TUCS

Tuomas Poikela

Readout Architecture for Hybrid Pixel Readout Chips

Turku Centre *for* Computer Science

TUCS Dissertations
No 198, June 2015

Readout architectures for hybrid pixel detector readout chips

Tuomas Poikela

To be presented, with the permission of the Faculty of Mathematics and Natural Science of the University of Turku, for public criticism in Auditorium Beta on June 15, 2015, at 12 noon.

University of Turku
Department of Information Technology
FI-20014 Turun yliopisto

2015

Supervisors

Adjuct Professor Juha Plosila
Department of Information Technology
University of Turku
FI-20014 Turun yliopisto
Finland

D.Sc. (Tech.) Tomi Westerlund
Department of Information Technology
University of Turku
FI-20014 Turun yliopisto
Finland

Reviewers

Professor Timo D. Hämäläinen
Department of Pervasive Computing
Tampere University of Technology
PL 527, 33101 Tampere
Finland

Professor Thomas Hollstein
Department of Computer Engineering
Tallinn University of Technology
Akadeemia tee 15A, 12618 Tallinn
Estonia

Opponent

Professor Angelo Rivetti
Department of Physics
The Torino University
Via P. Giuria, 1 - 10125 Torino
Italy

ISBN 978-952-12-3235-0
ISSN 1239-1883

Abstract

The original contribution of this thesis to knowledge are novel digital readout architectures for hybrid pixel readout chips. The thesis presents asynchronous bus-based architecture, a data-node based column architecture and a network-based pixel matrix architecture for data transportation. It is shown that the data-node architecture achieves readout efficiency 99% with half the output rate as a bus-based system. The network-based solution avoids "broken" columns due to some manufacturing errors, and it distributes internal data traffic more evenly across the pixel matrix than column-based architectures. An improvement of $> 10\%$ to the efficiency is achieved with uniform and non-uniform hit occupancies.

Architectural design has been done using transaction level modeling (TLM) and sequential high-level design techniques for reducing the design and simulation time. It has been possible to simulate tens of column and full chip architectures using the high-level techniques. A decrease of > 10 in run-time is observed using these techniques compared to register transfer level (RTL) design technique. Reduction of 50% for lines-of-code (LoC) for the high-level models compared to the RTL description has been achieved.

Two architectures are then demonstrated in two hybrid pixel readout chips. The first chip, Timepix3 has been designed for the Medipix3 collaboration. According to the measurements, it consumes < 1 W/cm^2. It also delivers up to 40 Mhits/s/cm^2 with 10-bit time-over-threshold (ToT) and 18-bit time-of-arrival (ToA) of 1.5625 ns. The chip uses a token-arbitrated, asynchronous two-phase handshake column bus for internal data transfer. It has also been successfully used in a multi-chip particle tracking telescope.

The second chip, VeloPix, is a readout chip being designed for the upgrade of Vertex Locator (VELO) of the LHCb experiment at CERN. Based on the simulations, it consumes < 1.5 W/cm^2 while delivering up to 320 Mpackets/s/cm^2, each packet containing up to 8 pixels. VeloPix uses a node-based data fabric for achieving throughput of 13.3 Mpackets/s from the column to the EoC. By combining Monte Carlo physics data with high-level simulations, it has been demonstrated that the architecture meets requirements of the VELO (260 Mpackets/s/cm^2 with efficiency of 99%).

Tiivistelmä

Tässä tutkimuksessa analysoidaan uusia digitaalisia tiedonsiirtoarkkitehtuureita hybridipikseli-ilmaisimien lukupiireille. Väitöskirja esittelee asynkronisen väylän, data-solmuihin perustuvan pystyriviarkkitehtuurin sekä verkkopohjaisen arkkitehtuurin pikselimatriisin lukuun. Tutkimuksessa näytetään miten solmupohjaisella arkkitehtuurilla voidaan saavuttaa 99% tiedonlukutehokkuus käyttäen matalampaa nopeutta kuin väyläpohjaisessa arkkitehtuurissa. Verkkopohjainen ratkaisu puolestaan lisää sietokykyä rikkinäisiä pikselipystyrivejä vastaan, ja se jakaa piirin tietoliikenteen tasaisemmin pikselimatriisin sisällä kuin pystyrivipohjaiset ratkaisut. Yli 10% parempi lukutehokkuus on saavutettu verrattuna pystyriviarkkitehtuuriin.

Arkkitehtuurin suunnittelu on tehty käyttäen TLM- ja sekventiaalisia korkean tason suunnittelutekniikoita. Käyttäen näitä menetelmiä on pystytty simuloimaan kymmeniä erilaisia pikselipystyrivi- ja koko piirin kattavia arkkitehtuureita. Simulaatioiden ajoaika on lyhentynyt yli kymmenkertaisesti näillä tekniikoilla verrattuna RTL-suunnittelutekniikkaan. Korkean tason mallit ovat noin 50% kompaktimpia koodiriveissä laskettuna.

Kahta näistä arkkitehtuureista esitellään tarkemmin kahdessa eri piirissä. Ensimmäinen piiri, Timepix3, on suunniteltu Medipix3-kollaboraatiolle. Mittausten mukaan tehon kulutus on < 1 W/cm^2. Piirin lukunopeus on 40 Mpikseliosumaa/s/cm^2, joista jokainen sisältää 10 bitin varaustiedon sekä 18 bitin aikatiedon 1.5625 nanosekunnin tarkkuudella. Piiri käyttää asynkronista kaksivaiheista väyläprotokollaa sisäiseen tiedonsiirtoon. Sitä on myös onnistuneesti käytetty useammasta piiristä rakennetussa ilmaisimessa hiukkasten jäljittämiseen.

Toinen piiri, VeloPix, on Euroopan ydintutkimuskeskuksen CERNin LHCb-kokeen VELO-ilmaisimen päivitystä varten kehitetty pikselilukupiiri. Simulaatioiden perusteella piirin tehon kulutus on < 1.5 W/cm^2. Piirin lukunopeus on 320 Mpakettia/s/cm^2, joista jokainen sisältää jopa 8 pikseliosumaa. VeloPix käyttää solmupohjaista arkkitehtuuria saavuttaakseen lukunopeuden 13.3 Mpakettia/s pikselipystyriviltä. Yhdistämällä Monte Carlo simulaatioden tulokset korkean tason simulaatioihin on näytetty, että arkkitehtuuri täyttää VELOn vaatimukset (260 Mpakettia/s/cm^2 99%:n tiedonlukutehokkuudella).

Acknowledgements

Although writing a thesis and getting your thoughts down on paper is sometimes a lonely road, I could not have done all the work presented here without support from family, friends and colleagues.

First and foremost, I would like to thank my supervisors Tomi and Juha at Turku for giving me this opportunity to pursue my PhD at the university of Turku, and giving me freedom to carry out the research in Switzerland at CERN. They have supported me during these years and always given me advice when needed.

I am grateful to Ken for supervising me at CERN during my thesis work, and giving me a chance to join the working community of CERN in the first place. I also want to thank him for motivating me to write when other projects seemed more interesting.

Professors Thomas Hollstein and Timo D. Hämäläinen are gratefully acknowledged for reviewing this thesis, and professor Angelo Rivetti for taking the time to be my opponent in the defense of this thesis. I also gratefully acknowledge that this thesis was funded by Turku Centre for Computer Science (TUCS) and the European Organization for Nuclear Science (CERN) Doctoral student program. EURO-DOTS is also acknowledged for giving me scholarship for PhD courses.

I would like to thank my colleagues in CERN ESE group for many lively discussions on electronics and topics completely unrelated to work as well. Thanks to Michael and Xavi for trusting in my skills and giving me the chance to work in Timepix3 project as part of my PhD. Also thanks to all the people in Medipix3 collaboration for being open to my new ideas about Timepix3. Thanks to Timepix3 readout system development team, and especially Szymon for the on-site support. Thanks to Erik for providing me Python scripts. Thanks to Jerome for helping with the Timepix3 measurement hardware. Thanks also to other colleagues in the CERN Medipix team: Massimiliano, Rafa, Winnie, Pierpaolo. Thanks to all other designers of Timepix3 for all the hard effort to make the chip work: Vladimir G, Yunan, Francesco. Thanks to Matt for many discussions on digital design. Thanks to colleagues in the LHCb VELO, especially Martin, Paula and Jan for trusting me with the VeloPix design. Thanks to Jonne for sending me his thesis from Finland.

Thanks to all the people who have made my stay in Geneva very enjoyable and have shown there's more to do than just work: climbing buddies, bandmates, other friends and especially ”perhetutut”.

Thanks to my parents Maarit and Timo for their encouraging words and support throughout my life.

Above all, I am in gratitude to my wife Hanna for her support and companionship during my doctoral studies. And finally special thanks to our daughter Emilia who has brought so much joy into my life.

Geneva, May 2015

-Tuomas Poikela

Contents

List of Figures

List of Tables

List Of Acronyms

10GbE 10 Gb Ethernet

ADC analog-to-digital converter

AFSM asynchronous finite state machine

ALU arithmetic logical unit

API application programming interface

APS active pixel sensors

ASIC application specific integrated circuit

ATLAS A Toroidal LHC apparatuS

BX-ID bunch crossing identification data

CC Cyclomatic Complexity

CCD charge-coupled device

CERN the European Organization for Nuclear Science

CMA cumulative moving average

CMOS complementary metal oxide semiconductor

CMS Compact Muon Solenoid

DAC digital-to-analog converter

DC double column

DDR double-date rate

DICE dual interlocked cell

DSP digital signal processor

DVFS	dynamic voltage and frequency scaling
ECC	error correction coding
EDA	electronics design automation
ELT	enclosed layout transistor
ENC	equivalent noise charge
EoC	End of Column
FF	fast corner
FIFO	first-in first-out
FSM	finite state machine
FPGA	field-programmable gate array
GaAs	gallium arsenide
GALS	globally-asynchronous locally-synchronous
GBE	gigabit ethernet
Ge	germanium
HDL	hardware description language
HPD	hybrid pixel detector
IC	integrated circuit
IO	input-output
IP	intellectual property
ISS	instruction-set simulator
LFSR	linear-feedback shift register
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LoC	lines-of-code
LQF	longest queue first
LSB	least significant bit

LUT lookup-table

LWF Longest-Wait-First

MAPS monolithic active pixel sensor

MBU single-event multiple-bit upset

MC Monte Carlo

MSB most significant bit

NoC network-on-chip

NRE non-recurring engineering

OCF Oldest-Cell-First

OCV on-chip variation

OVM Open Verification Methodology

PCB printed circuit board

PLL phase-locked loop

PUC pixel unit cell

PnR place and route

PVT process, voltage and temperature

RAM random access memory

RICH ring imaging Cherenkov

RNG random number generator

RTL register transfer level

RX receiver

SCPI standard commands for programmable instruments

SDC Synopsys design constraint

SDF standard delay format

SEE single-event effect

SET single-event transient

SEU	single-event upset
SLVS	scalable low-voltage signaling
SNR	signal-to-noise ratio
Si	silicon
SP	super pixel
SPIDR	Speedy PIxel Detector Readout
SPEF	standard parasitic extraction format
SPP	super pixel packet
SRAM	static random access memory
SS	slow corner
STA	static timing analysis
STI	shallow trench isolation
SV	SystemVerilog
TDC	time-to-digital converter
TID	total ionising dose
TLM	transaction level modeling
ToA	time-of-arrival
ToT	time-over-threshold
TT	typical corner
TX	transmitter
TMR	triple modular redundancy
TSV	through-silicon via
UVM	Universal Verification Methodology
VCD	value-change-dump
VCO	voltage-controlled oscillator
VELO	Vertex Locator

VHDL very high-speed integrated circuit hardware description language

V$_t$ threshold voltage

VLSI very large scale integrated

WRR weighted round-robin

Chapter 1

Introduction

Imaging sensors are being integrated into many applications due to CMOS scaling and cheapening manufacturing of integrated electronics. One type of sensor, a pixel sensor, is used in digital cameras [1] and applications requiring dynamic vision [2]. Other fields like dosimetry [3] and medical imaging [4] are starting to use CMOS circuits more widely than before. Other emerging applications for pixel sensors are artificial retina prostheses [5], for example.

A typical readout chain for a pixel detector system is shown in Figure 1.1. In pixel sensors, incident radiation, such as X-ray photons create small charge signals in a sensitive volume sub-divided into regularly-spaced elements called pixels. The positional information of the incident particle is thus given by the address of the pixel(s) containing signal. This allows the formation of an image. Enhancements can be made using, for example, the amplitude of the signal or counting the number of signals during exposure.

Pulses from a sensor are amplified and digitized using a dedicated readout chip. In the case of a monolithic detector, the same chip functions as a sensor and a readout chip. Due to increasing demands for more information per pixel and higher signal rates together with strong constraints on power consumption, an efficient architecture is required to extract the digitized data. *This thesis looks for novel solutions for transporting data from pixels to the output of the readout chip within an imaging sensor readout application specific integrated circuit (ASIC).* The focus is solely on the internal readout architecture of a readout ASIC. As an additional restriction, these architectures must be able to provide timing information with sufficient accuracy attached to all data while keeping the pixel size as small as possible. Each digitized pixel hit can consist of address bits only, for example 16 bits per hit for 256×256 pixels, or it can contain digitized time and charge information in addition, for example. This can increase the total number of bits per digitized hit to over 50.

After the digitized data has been transported from the pixels to the

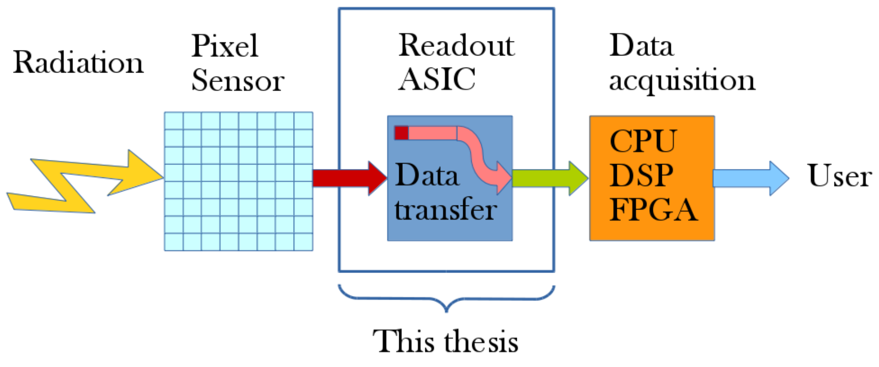


Figure 1.1: A readout chain of a pixel detector.

output of the readout chip, it is transmitted off-chip for further processing. This processing can be done by a field-programmable gate array (FPGA) or a digital signal processor (DSP) or even a standard off-the-shelf microprocessor. It is also possible to connect multiple sensor/readout ASIC pairs to one data acquisition system. Finally, the data is presented to a user of the application for analysis or inspection. The user can be either another machine or a human. An example of the first one is an automatic alarm system which makes a decision to sound an alarm based on the data coming from a data acquisition system, which in turn receives its data from a readout chip connected to a light-sensitive sensor.

1.1 Applications of pixel detectors

Two possible applications for pixel detectors are shown in Figure 1.2. In tracking applications, detectors are used for measuring the position and time when a particle passes through a pixel. Track reconstruction always requires more than one plane of pixels. In Figure 1.2, on the left side two different tracks captured by pixel detectors are shown. Tracks A and B can occur at different times or at the same time. By recording time information relative to a common reference in addition to pixel coordinates, tracks can be correlated with particular discrete events in time. These tracks can originate from an event which is typically a collision of particles in a high-energy physics experiment, such as protons, or atoms such as lead. Incident particles registered by tracking detectors are usually decay products of other particles, thus tracking information is used for reconstructing the patterns of decay sequences.

On the right side of Figure 1.2, an imaging application using a pixel detector is shown. The detector is used to capture energy information about

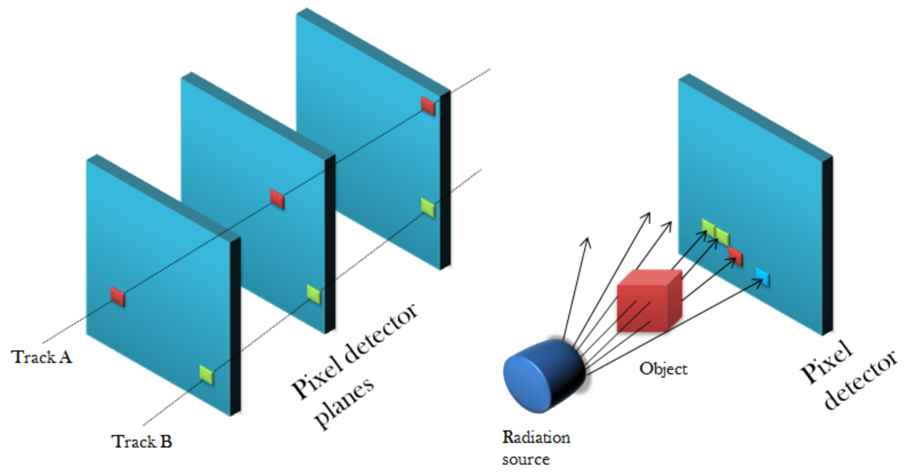


Figure 1.2: Two applications of pixel detectors: Tracking (on the left) and object imaging (on the right).

particles emitted by a radiation source. An object between the source and the detector absorbs part of the energy of a particle depending on the thickness and material of the object. Pixels shown in different colors will capture different amount of charge, and this charge information can be used to reconstruct the image. Using this method, it is possible to determine material inside the object, for example. One of the main differences between tracking and imaging applications is, that while tracking focuses on individual particle tracks, in imaging the image can be formed by integrating several tracks in a single pixel. Thus, in a tracking application, it can be beneficial to transfer hit data off a readout ASIC as quickly as possible, whereas in an imaging application data can be accumulated at the pixel-level for longer periods of time.

Pixel detectors have been used for tracking at the European Organization for Nuclear Science (CERN) in large applications such as Compact Muon Solenoid (CMS) [6] and A Toroidal LHC apparatuS (ATLAS) [7]. Large Hadron Collider beauty (LHCb) also uses pixel detectors in its ring imaging Cherenkov (RICH) detector [8], and pixels are being investigated as a detector option for the VELO detector [9] of the LHCb upgrade [10]. These applications typically require tracking precision down to a few microns and pixel sizes of a few tens of microns. As these applications operate in an environment with radiation levels orders of magnitude higher than background radiation, the choice of a type of pixel detector is also important, as will be discussed later. This thesis focuses solely on readout architectures of ASICs used in tracking applications.

1.2 Hybrid pixel detectors

The focus of this thesis is on digital data readout architectures of ASICs for a particular type of pixel detectors called hybrid pixel detectors (HPDs). An HPD consists of two distinct chips called a sensor and a readout chip. Both chips can be manufactured using different processes and optimized separately, with the readout chip being generally fabricated in standard CMOS process. The sensor chip is also a solid-state device, manufactured using a semiconductor as the sensor material. Typical materials for sensors are silicon (Si), germanium (Ge), gallium arsenide (GaAs) and diamond. Semiconductor materials are self-supporting structures (unlike gas sensors, for example), have an average energy of 3.6 eV for creating an electron-hole pair and the signal is collected in the order of 10 ns [11].

A cross-section of an interconnected sensor and a readout ASIC is shown in Figure 1.3. When a charged particle passes through the sensor, it creates electron-hole pairs inside the sensor. The number of pairs depends on, for example, the ionisation energy of the material, the energy of the incident particle and the length of its path in the material. By applying a bias voltage across the sensor, an electric field is created which causes electrons to drift from lower potential to higher. Respectively, it causes holes to drift towards lower potential and then being collected by the $p+$ -region. As can be seen from Figure 1.3, a p-n diode is used for collecting the signal inside the sensor. It is a reverse-biased diode with fully depleted region. Typical signal magnitude is 23000 electrons for a silicon of 300 μm [12]. Another value given in the literature is 20000 electrons or holes per 250 μm in a fully depleted silicon sensor, corresponding to an input charge of about 3 fC [13].

Multiple pixels can receive a signal from one particle if it crosses several pixels due to low incident angle. This effect, which creates a cluster of pixels associated with one particle only, is called charge sharing. Charge sharing is useful in tracking applications for finding a more precise location of the track. By measuring the amount of charge in each pixel of the cluster and taking, for example, center of the mass of these charges, improved spatial resolution can be achieved. Without any charge information, the resolution is given by $\frac{p}{\sqrt{12}}$ [12] for square pixels with a pitch of p. The drawback of charge sharing is that the signals per pixel are smaller in amplitude because the charge is split among several pixels and hence are more difficult to detect.

The p-n diode structure inside the sensor is connected to an interconnection between the two chips. This interconnection between the sensor and the readout chip is also called a bump-bond, and the chips are connected together using a solder process called bump-bonding [14]. The bump is connected to the readout ASIC and ultimately to the front-end electronics via the full CMOS metal stack. The purpose of the front-end electronics is to amplify and digitize signals generated by charged particles in the sensor.

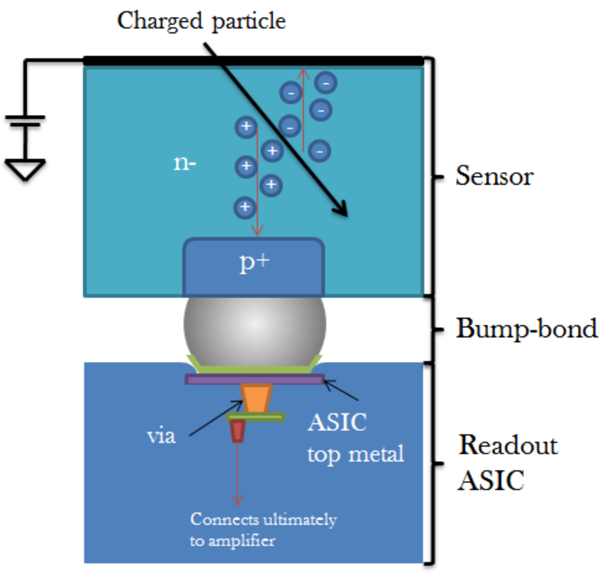


Figure 1.3: A cross-section of a sensor and a readout chip connected with a bump-bond.

Figure 1.4 shows an example of a hybrid pixel detector of multiple chips. As mentioned, the sensor chip is mounted on top of the readout chip using bump-bonds. These bumps form the electrical connection between the two chips. Connections from readout chips to a readout system are omitted for clarity. These connections can be made using a technique called wire-bonding or by using connections on back-side of the chip by deploying a redistribution layer [15].

A readout chip of a pixel sensor typically has a pixel area called the pixel matrix and a peripheral area. This division is shown in Figure 1.5. The sensor is located on top of the active area and the periphery extends over the sensor edges. Pixels inside the readout chip are occupied always by analog signal processing functions and often digital logic. Analog processing is required to convert signals from a sensor to full CMOS voltage levels. The area available for electronics is constrained by the sensor pixel size. This often introduces conflicting requirements between tracking precision (smaller pixels are better) and functionality (larger pixels allow more electronics).

1.2.1 Noise

Both the sensor and the readout ASIC introduce noise into the system. A leakage current in the sensor causes a signal to be generated even in the absence of an incident particle. The most important contribution to the leakage current is given by thermal generation at the surface of the device

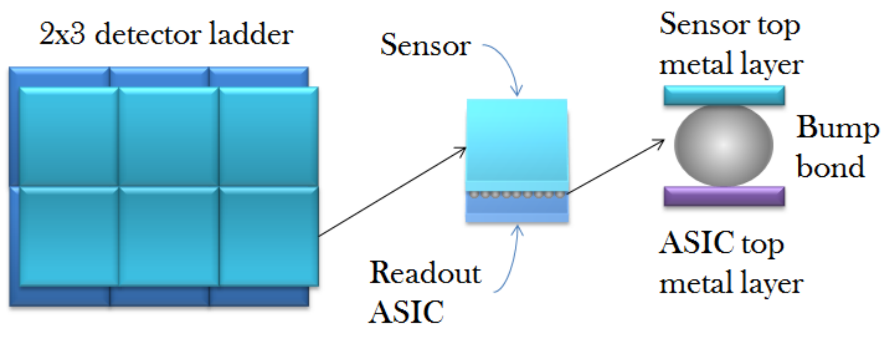


Figure 1.4: An HPD ladder consisting of 2 × 3 sensor and readout chips.

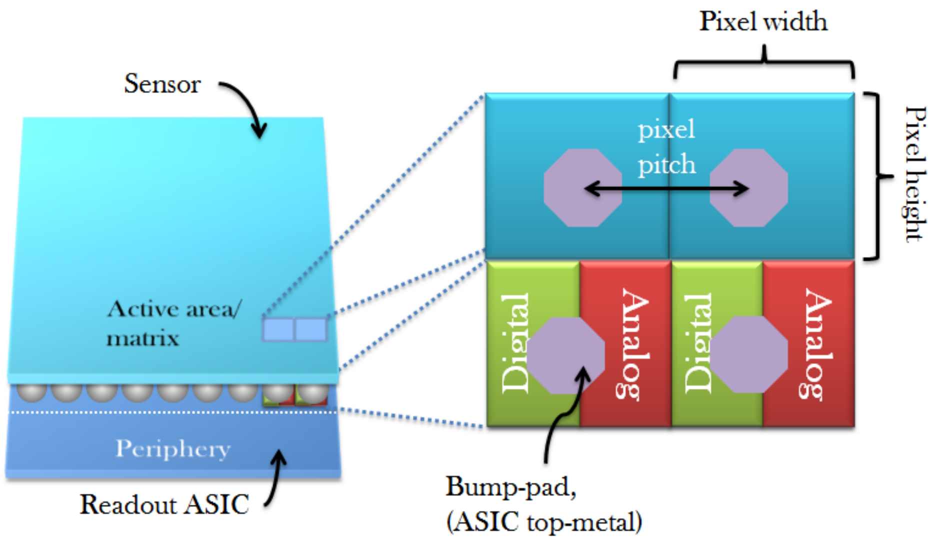


Figure 1.5: Periphery and active area of a hybrid pixel sensor and a close-up of two sensor pixels and two readout pixels.

and in the depleted volume of the sensor [12].

The leakage current of the sensor adds shot-noise to the signal which is presented to the readout electronics. A leakage current compensation circuitry can be implemented inside the readout ASIC by placing a current source or sink, depending on signal polarity, between input and output of a low-noise amplifier in the front-end [12].

The readout ASIC itself has several sources of noise. Cross-talk from digital logic to analog is a contributor to the noise of the analog front-end. This can be caused by direct capacitive coupling of frequently switching digital signals or indirect coupling via digital power supply or ground bounce through a silicon substrate. Because signals that can be detected by the analog front-end may be only 400-500 electrons in magnitude [16] or even smaller, the analog front-end is very sensitive to noise coming from the digital logic. In fact, extra noise injected into the analog front-end will increase the minimum detectable charge. The extra noise manifests itself as extra pixel hits which are not caused by signals coming from a sensor but from the readout ASIC itself. A typical measurement unit of noise for HPD systems is equivalent noise charge (ENC) which indicates a point where signal-to-noise ratio is equal to 1 [17].

For timing measurements, the rise time of a front-end amplifier combined with the signal-to-noise ratio determines the timing jitter of the system [17]. A quantization error occurs when the analog signal is converted into a discrete, digital value. This error is simply the difference of the actual analog value and the digitized value. This error is present, for example, when charge of a particle or time of arrival are measured using a sampling clock. The clock distribution itself also contributes to the quantization error because the clock signal cannot be distributed across the full chip in zero time thus arriving to different pixels at different times.

1.2.2 Radiation tolerance

HPDs are often used in environments with significant levels of background radiation such as particle physics experiments. This radiation can potentially affect the readout and sensor chips. Radiation effects in electronics are usually divided into total ionising dose (TID) and single-event effects (SEEs). TID is an accumulating effect which becomes worse the longer a device is exposed to ionising radiation. The leakage current of the device increases due to charge trapped inside the shallow trench isolation (STI) oxide. Even if the device is turned off, this charge can create a leakage current path from drain to source. For 130 nm CMOS technology, this current has been measured to be less than 1 μA per transistor [18]. The current driving capability of transistors decreases, partly due to an increase in the threshold voltage (V_t) of transistors, partly due to decrease in transconductance of transistors [19]. A study on a 65 nm CMOS technology shows that PMOS transistors are particularly vulnerable to such an effect [19]. However, effects of radiation in 130 nm CMOS are well-understood and the technology shows an improvement in radiation tolerance when compared to older technologies [18]. In this thesis, 130 nm CMOS technology is used for implementing circuits. As discussed in [20], a commercial 130 nm CMOS technology is sufficiently tolerant against TID effects even when using linear transistors instead of enclosed layout transistors (ELTs) which take up more area. ELTs are typically used to improve the radiation tolerance of transistors.

An SEE is the result of an instantaneous impact of radiation affecting the state of the electronics, and can occur either as a single-event transient (SET) or a single-event upset (SEU). The former causes a transient change of voltage in one of the capacitive nodes of a logic gate or a memory cell. The likelihood of an SET decreases with increasing node capacitance. If this change is captured by a memory device, it becomes a persistent effect. On the other hand, an SEU directly causes a memory element such as a flip-flop to invert its state. Unless the device is self-correcting, a new state will persist until a new value is written into the memory device. The new state will also propagate to all the logic connected to the fan-out of this device. Mitigation techniques for SEUs will be discussed later in this thesis. Other effects such as single-event latch-up or transistor gate rupture can also be caused by radiation. However, no evidence of gate ruptures has been observed in 130 nm CMOS [21] during irradiation.

Bonacini *et. al.* [22, 19] have studied SEUs in 65 nm and 90 nm CMOS technologies. It has been concluded that the probability of an SEU in a single device decreases as transistor size is decreased. Although smaller devices have less capacitance, the probability of hitting a sensitive node in the device is also smaller. On the other hand, the number of devices on a single chip also increases so the probability of the SEUs across the whole system does not decrease or increase significantly due to CMOS scaling.

Design techniques such as triple modular redundancy (TMR) [23] and error correction coding (ECC) can be used to make circuits very tolerant to SEEs. TMR is based on triplicated logic in which the correct result is a vote of the three outputs. If only one device has been upset, the output of the voting is still correct. ECC such as Hamming coding can also be used to correct single-event upsets or even detect multiple bit upsets. These techniques, however, introduce area, power and timing penalties. Veeravalli [24] reports 202% - 208% area overhead when using TMR for a 32-bit arithmetic logical unit (ALU). It also reports 148% overhead for an ALU using ECC. Generally it can be concluded that for a fully triplicated design, the area overhead is always more than 200% as voting logic is required in addition to the triplication overhead. Another useful property of these techniques is that they can also be used to improve the yield. In a case where the chip has a manufacturing error in one of the protected nodes or memories, the logic will still function correctly but the SEE tolerance is lost.

1.2.3 Power

Optimization of power in an HPD readout ASIC is important for two following reasons. Firstly, the material placed in front of a detector needs to be minimized to distort the measurement as little as possible. When heavy materials are placed on the tracks of particles, part of their energy gets absorbed by the material and their trajectory can be perturbed. This prevents usage of large heat sinks for cooling, for example. Additionally, if the temperature of the readout ASIC increases, this increases leakage current and electromigration within the chip. This may make the chip slower and decrease the life-time of the device.

Secondly, due to geometry of the readout ASIC, power is brought into the chip from the periphery. Power distribution for the pixel matrix must be done from the periphery to the top of the chip. This results in long metal wires having significant resistance. For example, a 1.4 cm long copper line in 130 nm CMOS with a width of 25 μm has a resistance of around 4 ohms. This puts a limit on the maximum amount of static and dynamic current that can be drawn before the voltage drops have an impact on the operation of transistors.

The power consumption of the readout ASIC can be divided into analog and digital power consumption. Most of the power consumed by analog electronics is typically static power [25]. There are also architectures with dynamic components [26], mainly dynamic comparators that can be clocked. Using more power at the analog front-end makes the front-end faster and improves the timing resolution, but it also introduces more noise into the front-end.

The digital power consumption is a sum of leakage power (static) and switching activity (dynamic). Each of these elements can be optimized independently of each other, and the optimization depends on operation conditions such as temperature and the expected activity in the application. The leakage can be reduced by decreasing the power supply, using transistors with higher V_t and shutting down the power completely (power gating) from unused parts of the chip. Typical methods for controlling the dynamic power consumption are clock gating, operand isolation and dynamic voltage and frequency scaling (DVFS). The first two are supported by electronics design automation (EDA) tools when using RTL design methodology, while the latter usually required manual implementation. So far, no DVFS has been deployed in hybrid pixel readout chips.

Minimizing the switching capacitance by avoiding long metal wires that constantly change their value also reduces the dynamic power consumption. The architectural choices also have a large impact on the dynamic power consumption. A simple example of this is a reduction of a 64-bit bus to a 16-bit bus and sending four words in one transaction of four clock cycles instead of one transaction of one clock cycle. This effectively reduces the instantaneous power consumption by four (on average) but decreases throughput of the bus. Note that this method does not save any energy.

In applications requiring low duty cycles, power gating (also known as power pulsing or power cycling) can also be used to reduce the power consumption. In this scheme, electronics are switched into a lower power state by either switching off the power supply or by altering the bias voltages of transistors to reduce their current consumption. The former option is especially useful for digital logic in which no state information needs to be saved. The latter option is more suitable for analog electronics because it does not increase additional transistors and thus additional noise to power supply lines.

1.2.4 Other types of pixel detectors

In addition to HPDs, there are other types of pixel detectors. Passive detectors like charge-coupled devices (CCDs) contain no active electronics and are not discussed in this thesis. Like HPDs, they use solid-state sensors for collecting the charge.

Another type of active detectors besides HPDs is the monolithic active pixel sensor (MAPS). Unlike HPDs, they consist of a single chip only, where it functions as a sensor and a readout chip at the same time. A cross-section of a MAPS detector is shown in Figure 1.6. The charge is collected by the n-well diode. The sensor is only fully depleted under this diode, and the charge collection is incomplete elsewhere in the epitaxial layer [12]. In addition to charge collection, they contain signal processing functionality, typically an amplification and digitization of signals. MAPSs cannot be implemented using standard CMOS processes, and require additional processing steps, for example triple wells, if CMOS logic is used.

Integrating the sensor and the readout electronics into the same ASIC is an advantage in terms of cost compared to HPDs. In MAPS, pixels can be implemented with as few as three transistors per pixel and the pixel pitch can be smaller than 10 μm but have smaller signal-to-noise ratio (SNR) than HPDs [27]. The noise is produced by a phenomenon called dark current [28].

A typical digital camera found in cell phones is usually implemented as a MAPS because it requires smaller pixels, has to be cheap to manufacture and requires only frame readout rates of the order of few kHz. The readout rates of MAPSs are in general limited by the CMOS technology and the relative simplicity of the circuitry inside a pixel. Layout techniques like

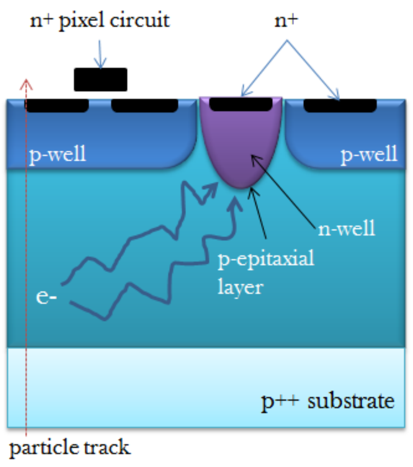


Figure 1.6: A cross-section of an MAPS showing the n+ -well diode on the p-epitaxial layer. [12]

triple-well can be used in MAPS to have the benefits of full-fledged static CMOS design at the expense of extra process layers and the extra area taken by the triple-well.

MAPSs are also less resilient to radiation effects than HPDs. This is an important attribute, particularly for high-energy physics experiments, where total dose of radiation can be hundreds of Mrads or even higher [29]. HPDs on the other hand are more expensive to manufacture than MAPSs because they require additional processing steps for connecting the sensor to the readout chip. Despite this cost, they have been deployed in applications where the requirements of readout speed and radiation tolerance cannot be met by MAPS.

1.3 Requirements

Figure 1.7 shows a set of performance metrics that need to be chosen for each readout architecture designed for tracking applications. Each parameter is shown with its typical unit. Generally, when performance in one category is improved, the expected performance in some other category deteriorates. For example, when the spatial resolution is improved (pixel size decreased), all other things being equal, power density and thus power consumption increases because power does not scale down with the pixel size. There are other metrics such as radiation hardness which typically have an impact on power consumption, chip area and indirectly to other metrics as well. The most relevant metrics for this thesis are readout rate, power, chip area, timing range and timing resolution. These will be discussed in more detail

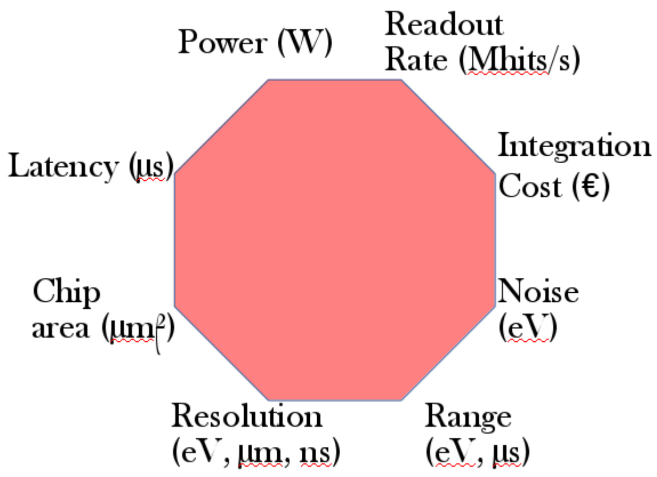


Figure 1.7: Different performance metrics for a pixel readout ASIC.

Table 1.1: General requirements for digital readout architectures.

Pixel size	55 $\mu m \times$ 55 μm
Number of pixels	256 $\times$ 256
Chip area	2 cm^2
Time resolution	at least 25 ns
Timing range	at least 9 bits
Latency	$<$ Timing range
Power	$<$ 1.5 W/cm^2
Readout rate	$>$ 40 Mhits/s/cm^2

particularly in Chapters 2 and 4.

Table 1.1 shows the general requirements for digital readout architectures presented in this thesis. The pixel and the matrix sizes are fixed to specific sizes but all the results presented later in this thesis are directly applicable to larger pixels sizes and smaller matrix sizes. Also, where mentioned, the results can be scaled to larger matrices as well. Time resolution must be always at least 25 ns with a range extending to at least 9 bits. Notice that the maximum latency here is simply the timing range. A detailed explanation of this will be given in Chapter 2. Due to limited cooling options in the hybrid pixel detectors, power consumption should be $<$ 1.5 W/cm^2. Finally, to improve readout rates compared to existing architectures, a minimum rate of 40 Mhits/s/cm^2 is required. As will be seen in Chapter 6, a factor of 10 higher rates than this can be achieved.

1.4 Scope and original contributions

This thesis focuses on the study and optimization of readout ASICs of HPDs in tracking applications. The scope of the work is digital very large scale integrated (VLSI) design, simulation and implementation. It studies digital data transfer techniques and their optimization from the pixel matrix to the periphery. It also studies on-chip data reduction techniques to overcome the problems of limited output bandwidth. Three original contributions to knowledge of the architectural design of HPD readout ASICs are presented:

1. The first contribution is a study to compare performance and implementation issues of several readout architectures, existing and new ones. The conceptual ideas from this study have then been tested by the author using high-level methodologies and simulation methods such as TLM and C++ simulation models. A subset of these simulations have then been performed at RTL. The physical implementation details of these architectures such as area and power are then studied in detail to estimate the feasibility of using these architectural techniques on a readout ASIC.

2. The second original contribution of this thesis is a sparse readout architecture capable of measuring time of arrival and charge of particles using a pixel of 55 μm $\times$ 55 μm for a mixed-signal chip called Timepix3 using 130 nm CMOS technology. In this chip, analog and digital elements are distributed uniformly across a chip of 1.4 cm $\times$ 1.4 cm, analog occupying 30 % and digital logic 70 % of the area. The author has designed and implemented the digital logic of the pixel and the super pixel. This includes all logic, buses and signal distribution required to transport data from the pixel matrix to the periphery as well as clock and time stamp distribution in the column. For the synchronizer in the digital front-end, the author has taken an existing design and adapted it to fit into new application. The author has also designed End of Column (EoC) block and periphery data bus architecture and arbitration for this chip. This chip has successfully been manufactured in silicon and its performance assessed. A comparison has been made between the simulated performance and the measurements.

3. The third contribution is the conceptual design, analysis and implementation of an architecture at the post-layout level for an HPD readout chip called VeloPix using 130 nm CMOS process. The author has designed a novel architecture for transporting data from the pixel matrix into the periphery. The periphery has also been designed to

sustain rates of over 1 Ghits/s/chip. The major challenges in addition to limited area available are the unprecedented data rates for HPD readout ASICs and the requirement of SEE tolerance which has been addressed using TMR techniques. This solution is capable of delivering more than 850 Mhits/s/cm^2 from a pixel matrix, or 320 Mpackets/s/cm^2 where each packet can contain 23 bits of information. This chip also features a novel solution for equalizing hit traffic from columns before they are sent off the chip.

Contributions presented here are based on RTL digital design principles to make them transferable to newer CMOS technologies more easily than custom circuit implementations. This implies reduced design time and cost while offering digital simulation and timing analysis tools for the verification of the designs. Higher level techniques such as TLM and behavioral (non-synthesizable) modelling have also been used in the architectural studies to allow exploration of larger range of design parameters in shorter time.

Timepix3 has been designed between 2010-2013, the chip being submitted in 24th May 2013. The first 6 wafers were received at the end of August 2013. Apart from the author's contribution, several designers from CERN (Geneva, Switzerland), Nikhef (Amsterdam, Netherlands) and the University of Bonn (Bonn, Germany) have contributed to the implementation and ideas of the chip. The design of VeloPix started in 2013 after the submission of Timepix3. It is a joint effort between CERN and Nikhef to design a new hybrid pixel readout chip for the upgrade of VELO of the LHCb experiment at CERN.

The work discussed and presented in this thesis in based on and extended from the publications listed below:

1. T. Poikela, J. Plosila, T. Westerlund, M. Campbell, M. De Gaspari, X. Llopart, V. Gromov, R. Kluit, M. van Beuzekom, F. Zappon, V. Zivkovic, C. Brezina, K. Desch, Y. Fu, and A. Kruth. Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout. Journal of Instrumentation, 9(05):C05013, 2014. [30]

2. T. Poikela, J. Plosila, T. Westerlund, J. Buytaert, M. Campbell, M. De Gaspari, X. Llopart, K. Wyllie, V. Gromov, R. Kluit, M. van Beuzekom, F. Zappon, V. Zivkovic, C. Brezina, K. Desch, Y. Fu, and A. Kruth. Digital column readout architectures for hybrid pixel detector readout chips. Journal of Instrumentation, 9(01):C01007, 2014. [31]

3. T. Poikela, J. Plosila, T. Westerlund, J. Buytaert, M. Campbell, X. Llopart, R. Plackett, K. Wyllie, M. van Beuzekom, V. Gromov, R.

Kluit, F. Zappon, V. Zivkovic, C. Brezina, K. Desch, X. Fang, and A. Kruth. Architectural modeling of pixel readout chips Velopix and Timepix3. Journal of Instrumentation, 7(01):C01093, 2012. [32]

4. M. van Beuzekom, J. Buytaert, M. Campbell, P. Collins, V. Gromov, R. Kluit, X. Llopart, T. Poikela, K. Wyllie, and V. Zivkovic. Velopix ASIC development for LHCb VELO upgrade. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, (0):0, 2013. [33]

5. V. Gromov, M. van Beuzekom, X. Fang, A. Kruth, R. Kluit, F. Zappon, V. Zivkovic, M. Campbell, T. Poikela, X. Llopart, C. Brezina, and K. Desch. Development and Applications of the Timepix3 Readout Chip. page 046. 20th Anniversary International Workshop on Vertex Detectors, Rust(Austria), June 2011 [34]

6. M. De. Gaspari, J. Alozy, R. Ballabriga, M. Campbell, E. Frojdh, J. Idarraga, S. Kulis, X. Llopart, T. Poikela, P. Valerio, and W. Wong. Design of the analog front-end for the Timepix3 and Smallpix hybrid pixel detectors in 130 nm CMOS technology. Journal of Instrumentation, 9(01):C01037, 2014. [16]

7. P. Valerio, J. Alozy, S. Arfaoui, R. Ballabriga, M. Benoit, S. Bonacini, M. Campbell, D. Dannheim, M. De Gaspari, D. Felici, S. Kulis, X. Llopart, A. Nascetti, T. Poikela, and W. S. Wong. A prototype hybrid pixel detector ASIC for the CLIC experiment. Journal of Instrumentation, 9(01):C01012, 2014. [35]

1.5 Related work and background

This section briefly summarizes the previous work done in the architectures of HPD readout ASICs, and offers a short motivation why this study is relevant. The technical details of these works are discussed in the following chapters of this thesis where applicable.

1.5.1 Simulation studies

No extensive study and simulation of the HPD readout architectures has been published linking the architectural studies to physical VLSI circuit implementation details. Individual studies are usually targeted for a specific application only [36, 37, 38]. These studies do not include comparisons between different architectures. In [39], an architecture optimized for a specific occupancy and hit distribution is presented.

Some studies offer tools for the evaluation of different architectures, but do not study the actual architectures or link the physical implementation details to them [40, 41].

1.5.2 Pixel readout architectures

In [12], an overview of mostly triggered readout architectures for hybrid pixel detectors is given. This serves as a good starting point for this study, and these architectures can be modified to operate continuously without a trigger signal.

Spatial resolution, which is directly related to the pixel pitch p, is an important parameter in pixel applications but not the only property of interest. In the architectures presented in this thesis, micrometer-level spatial resolution is combined with temporal resolution of nanosecond-level. The last parameter is the number of pieces of information (the number of pixels hit) with a given spatial and temporal resolution that can be measured and transferred from the readout chip to the data analysis tools.

HPDs with the same or smaller pixel pitch of 55 μm have already been manufactured [25, 42, 43, 44]. However, these chips are either lacking in time measurement capabilities [42, 43, 44], or cannot do simultaneous measurement of time and charge [25]. Chips with the simultaneous measurement capability have bigger pixel size, and thus lack in spatial resolution [37, 45, 46].

Some previous architectures are lacking in timing resolution and spatial resolution, typically having a minimum timing resolution of 25 ns[47, 48, 49].

Simultaneous time and charge measurement has been implemented with a pixel pitch of 25 μm [35] in 65 nm CMOS. However, this chip is at the prototype stage spanning only 1.6 mm $\times$ 1.6 mm, and having short dynamic range for timing measurement (4 bits at maximum of 100 MHz) because it is targeted for one specific application.

None of the chips cited have a minimum timing resolution of 1.5625 ns, which is targeted by one of the architectures presented in this thesis, except [45] which has a timing resolution better than 100 ps, but it also has a pixel 30 times larger than in the architecture presented in this thesis. None of the readout chip architectures mentioned above can provide trigger-less, continuous information with a rate of 40 Mhits/s/cm^2 or higher while having the spatial and timing resolution mentioned above.

By doing this PhD study, some missing features of related works are addressed. However, it needs to be noted that the HPD readout ASICs are usually highly customized for a particular application and its requirements, so completely generic, "one-size-fits-all" solution is unlikely to be found for all pixel detectors. Even so, information and outcomes of this thesis will be useful for future studies and simulation of HPD readout ASICs. In the following chapters, the technical details of existing architectures are analyzed and new options are also given. The simulation methodology used in this thesis is also presented, and its advantages and disadvantages discussed. The thesis is concluded by a presentation of two readout chips and their architectures targeting specific applications. One of them has been manufactured in silicon and tested to be fully functional. The other architecture has been implemented at the layout-level but not manufactured yet.

Chapter 2

Hybrid pixel detector readout ASIC architectures

In this chapter, different concepts of readout architectures of HPD ASICs are described. First, architectural details of a generic chip are shown. Definitions are presented for several characteristics of HPD readout ASICs to allow comparison of different chip architectures. Analysis of different readout architectures using simulation is presented with the comparison of simulation results between the architectures. Simulations in this chapter have been performed using SystemC [50] and non-synthesizable high-level models (except where stated otherwise) to facilitate faster modelling and simulation than with RTL or gate-level techniques. TLM techniques [51] have been used to connect the models together. The author has carried all the modelling, simulation, verification and analysis work required for this chapter.

The novelties presented in this chapter are a time stamping method to reduce the routing overhead and switching activity caused by sending the time stamp up the pixel column, and two different pixel readout architectures. The first one consists of data nodes communication locally with each other and propagating the data down the column through registers in the nodes. The second one is an extension of the data nodes to a network in which data is sent either horizontally or vertically to a next node.

From the simulations and analysis design guidelines are drawn which will be used later in this thesis for the architectural design of specific HPD readout ASICs. At the end of the chapter, related work and existing HPD readout ASICs are briefly summarized based on the definitions laid out earlier in this chapter.

2.1 Architecture of hybrid pixel detector ASIC

Figure 2.1 shows a general architecture which can be tiled from three sides. This means that any $2 \times N$ sensor ladder can be constructed from it. The advantage of the possibility to tile multiple chips together is to minimize dead area between sensors while being able to construct larger sensitive surfaces than from a single chip. Because yield of the sensor and readout chips is related to the total area per chip, the largest possible chip size may not always be desirable in terms of yield. On the other hand, there are other considerations such as unit cost per area where larger chips may be beneficial.

The chip is divided into two distinct parts, namely a pixel matrix and a periphery. As can be seen, most of the area is taken by the pixel matrix. Many HPD readout ASICs have this kind of division into a pixel matrix and a periphery [47, 48, 49, 25, 37, 45, 35]. The matrix is also called an active area because it corresponds to the sensitive part of the sensor. Ideally, the area taken by the periphery would be eliminated altogether to have the full chip covered by active area and to use it in four-side detector tiling. This would require usage of vertical buried interconnections called through-silicon vias (TSVs) [52], and would require having all input-output (IO) functionality on the backside of the readout chip. Ultimately TSVs enable 3D-integration of separate readout ASICs for analog and digital electronics.

2.1.1 Pixel matrix

A pixel matrix contains pixel unit cells (PUCs) connected to a sensor using bump-bonding techniques. A generic PUC is also described in [12]. Each PUC contains an analog signal processing electronics and in most cases digital logic for measurements and reading out the measurement data. This pulse processing chain is illustrated by Figure 2.2. The analog front-end is used for amplifying electrical signals arriving from sensors. These signals are typically short current pulses. It can also contain some pulse shaping functionality, threshold calibration, digital-to-analog converter (DAC) functionality and a digitisation scheme such as an analog-to-digital converter (ADC) or a discriminator. The pulse shaping is utilized to shape the signal suitable for ADCs or a discriminator. Local DACs are used to minimize the pixel-to-pixel threshold variations due to mismatches arising from process variations. Although most of these components are usually integrated inside a single pixel, for example in [45] there is no digital processing inside the pixel. Because HPD ASICs are very area-critical designs, front-ends and sensor signal processing cannot always be fully decoupled from the readout logic. One important concept related to the analog front-end is pile-up, which means the accumulation of charge into the electronics. Every time a

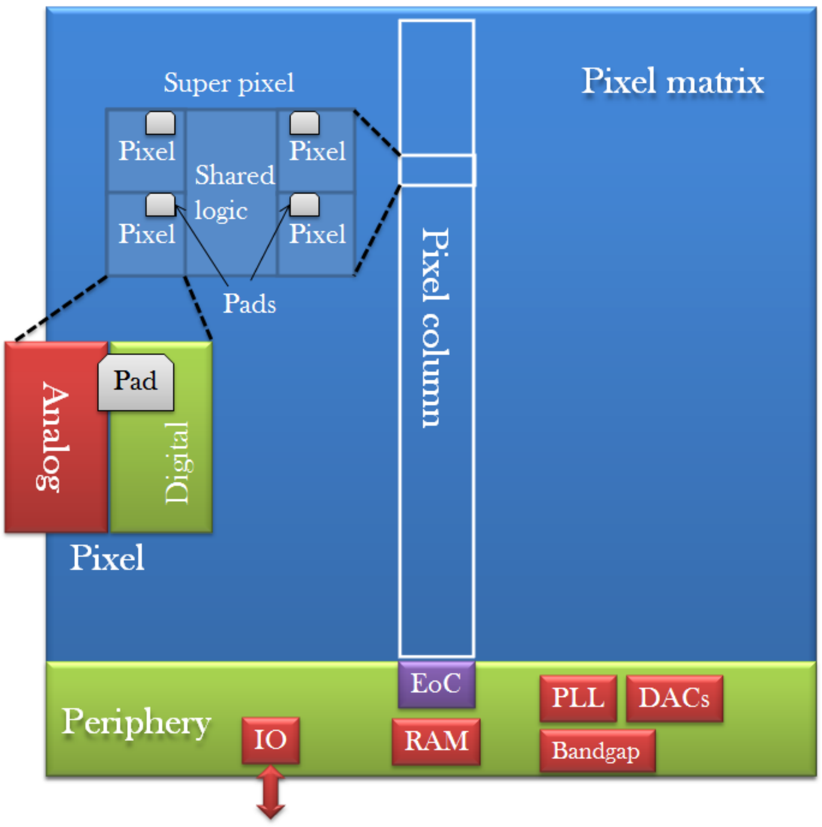


Figure 2.1: Structure of an HPD readout chip.

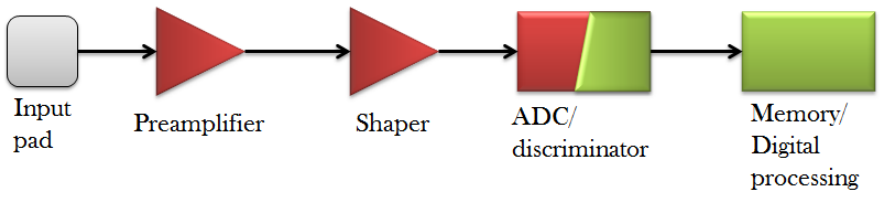


Figure 2.2: Pulse processing chain in an HPD readout chip.

signal arrives from a sensor, it carries a certain amount of charge. If the rate of the arrival of signals is high enough, the charge from the previous signal cannot be discharged completely. This leads to loss of charge information, and in the case of high rate, to a loss of all hit information.

The digital front-end typically contains synchronization logic, time-to-digital converters (TDCs), counters for measurements and memories for data storage and buffering. One essential function in chips utilizing a clock signal is to synchronize asynchronous hit events to the measurement clock to avoid glitches in counters that can cause errors in timing measurements. Upsets can arise if a gated clock signal has a glitch which is seen by some counter flip-flops but not all of them. The same situation occurs if an asynchronous hit signal is directly used as a counter enable signal, and is seen by only some flip-flops. Both cases lead to corruption of counter values and to incorrect measurement.

The digital front-end can also contain some intermediate buffering stages but the number of buffer slots is usually restricted by the area. While buffering can reduce the inefficiencies from high activity events, it also adds latency to the events before they are transmitted. Pile-up or hit overflow can also occur in digital electronics, if the digital front-end cannot store hit information from an event. This pile-up usually occurs due to dead-time in reading out the information from the pixel.

Several digital front-ends can share a common structure which is called a pixel region or a super pixel. These terms can be used interchangeably, and only the super pixel is used in this thesis. This grouping is also illustrated in Figure 2.1. A super pixel typically contains a common data buffer or memory, arbitration logic and some logic to interface with a column bus. In the case of a shift-register based column design, a super pixel can contain extra logic for zero suppression such as a hit-flag register [35]. If automated place and route (PnR) tools are used for laying out the super pixel, each digital front-end may have a different physical layout. This "flattening" of the design improves utilization and logic optimization between blocks but may cause mismatches in timing and differences in crosstalk between the digital and analog front-ends of the super pixel. These differences in crosstalk then manifest as variations in noise of analog front-ends.

Finally, pixels or super pixels are grouped together into a pixel column. Because the analog and digital front-ends are usually split into separate regions as shown in 2.1, the number of inter-column horizontal connections is usually very limited between two columns. This problem can be slightly alleviated by using a structure called a *double column* where the analog regions from two columns are placed adjacent to each other. All digital logic is then grouped between two analog double columns. By using a double column, bias voltages can be distributed to two columns using one metal line per voltage instead of two. Clock distribution across the double column can also be shared by two digital pixel columns. The main drawback of the double column is that input pads may have to be placed on top of a digital area thus making the pads more susceptible to crosstalk and noise injection from the digital logic.

2.1.2 Periphery

The periphery area of the chip consists of all peripheral blocks needed for a functioning HPD readout chip. The peripheral area has traditionally occupied a physical region of its own in readout chips outside the sensitive area. Future developments plan to remove this dead area using 3-D integration.

Regardless of the location of the periphery area, it contains global functionality, such as biasing circuitry and global configuration bits required in pixels. It provides input and output pads and interface logic to a readout system. This readout system can be another integrated circuit (IC) such as an FPGA or a complete desktop computer. Blocks such as DACs are used for the global biasing of the analog front-end circuitry. Bandgap reference circuits are utilized to provide temperature-independent voltages and phase-locked loops (PLLs) can be deployed to generate higher on-chip clock frequencies from an input reference clock. In large systems it is especially useful to be able to distribute a lower frequency clock and then generate the required high frequency on-chip clocks from this system clock. In some chips [49], random access memorys (RAMs) are used for buffering of data before sending it out of the chip. Typically, special radiation-hardened memory cells have to be used in applications in a high radiation environment. One chip provides an on-chip voltage regulator [37]. This makes the integration of the chip into a large system easier because it relaxes the requirements on the precision of the bulk power supply.

2.2 Readout ASIC data flow

Figure 2.3 shows the data flow inside an HPD readout chip. As mentioned, pixels are the first blocks to receive signals from the sensor chip. There is always a dead-time per pixel associated with processing of each arrival of

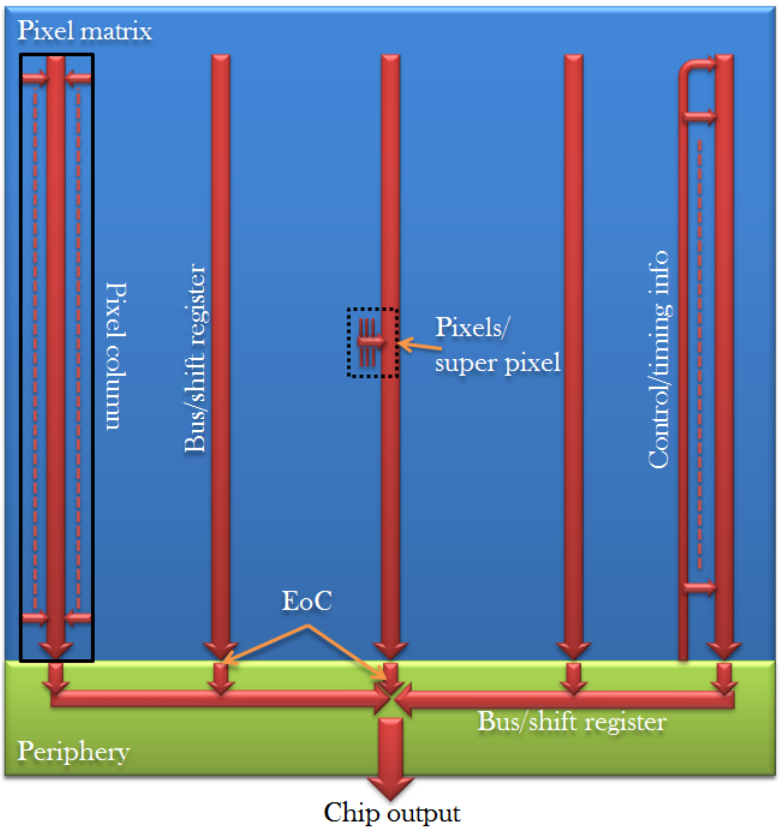


Figure 2.3: Data flow of an HPD readout chip.

a signal. This dead-time depends on the analog as well as on the digital front-ends. In the analog front-end, the dead-time is mainly determined by return-to-zero or return-to-baseline time which indicates when the analog front-end has discharged all signal received from the sensor and returned to its initial state before the signal. After this dead-time, a pixel has stored the measurement data either analogically or digitally, and this information must be read out. It can be done using a shift-register or a common bus between pixels.

If a chip uses the super pixel structure, data is transported from pixels to super pixels first. The super pixel then transmits data to the End of Column (EoC) block. This transmission can be done using shift registers [25, 35] or using either a digital or an analog data bus [53, 54, 49, 37, 46]. A third option is to drive the discriminator signals directly from the analog front-ends into the EoC [45]. This has an advantage of not requiring clock signals inside the pixel matrix for timing measurement but requires a high number of interconnections when the number of pixels increases.

In the periphery, data are sent from EoC to a chip output via a periphery bus or another suitable data fabric. This can be a network of data nodes communicating via handshake protocol, for example. In the periphery area there are usually more routing resources available for clock and signal distribution than in the pixel columns. The columns are limited in width by the pixel pitch, and large repeaters for signals may not fit into the column area. For this reason it may be possible to utilize a single parallel bus in the periphery running at higher frequency than in the columns. For example, if the column bus runs synchronously at 10 MHz and is 8 bits wide, the periphery bus could be run at 160 MHz with 64-bit parallel bus to provide sufficient bandwidth for 128 columns.

2.3 Measuring time

In tracking applications in which pixel hits are identified in time, timing reference signals or time stamps for pixel hits must be generated either on-chip or be provided externally. This generally takes the form of a counter(s) synchronized to a local or system clock combined with hit signals in the pixel. There are a number of different techniques each having its advantages and disadvantages. The timing reference can be calculated from a single signal or a timing reference bus can be distributed to pixels.

Figure 2.4 shows the two typical approaches for generating and registering the timing information. In Figure 2.4a, the timing reference is generated by a single global signal. When a pixel receives a hit, it starts an internal counter and counts until the global reference signal stops it. The drawback of this method is that the dead-time for calculating the time stamp can be up to $2^K - 1$ clock cycles where K is the number of bits in the time stamp. This dead-time can lead to inefficiencies and loss of data in the case of two or more hits arriving to the same pixel close enough in time. This kind of global timing reference is used for example in [25, 35]. The time stamp can also be derived from the address of the hit and added to the hit at the EoC if there is a known, fixed latency to transmit the hit from the column to the EoC [12].

Another, commonly used method is shown in Figure 2.4b. The timing reference is generated using a free running counter at the periphery of the chip. The counter value is then distributed to all pixels, and latched into a register at the arrival of a new hit. The dead time due to time stamp generation is negligible in this case because the time stamp value is ready one clock cycle after the rising edge of the discriminator. The drawback of this method is the distribution of a multi-bit bus across the full chip. This kind of time stamping is used in [47, 53, 49, 37, 46]. The time stamp bus is often Gray-encoded to reduce the number of transitions per clock cycle, and hence minimize the digital switching activity.

The timing reference signals must be distributed over the distance of a full column. For example, in [37] the distance is 336×50 μm = 16.8 cm. The capacitance C of wires in Figure 2.4 can be higher than several pF (0.2 fF / μm in 130 nm CMOS [55]) which requires strong drivers for driving the signals up the columns. The resistance R of the wires can exceed several $k\Omega$ (1 Ω / μm in 130 nm, 10 Ω / μm in 28 nm [55]) slowing down the signal propagation considerably. The difficulty of placing repeaters and eliminating the quadratic delay is usually related to the implementation of the bus inside a pixel column. In chips with hundreds of pixels per column, placing a repeater in every pixel cell would introduce an unacceptable insertion delay. Placing the repeaters less frequently breaks the pixel layout symmetry, and may introduce mismatch between pixels. To avoid this distribution, the time stamps can be also recorded at the EoC [54]. The drawback of this approach is that once the buffer for recording time stamps is full, all arriving hits are lost until a slot from the buffer is freed by reading the data out from all pixels associated with that time stamp.

A combination of the two approaches is also possible where dead-time is essentially traded-off for routing resources. It is possible to use a time stamp bus of $K - L$ bits, and use an L-bit counter inside a pixel. This counter is started at the arrival of a hit, and stopped with a global timing reference in the same manner as in Figure 2.4a. The difference is that the maximum dead-time is now $2^L - 1$ clock cycles instead of $2^K - 1$, where $L <= K - 1$. The problem of this approach is that a pixel must be read out within $2^L - 1$ clock cycles or ambiguities in time stamps will arise. This means that architectures with high throughput but also with high latency cannot utilize this method.

A novel method for time stamping is presented in Figure 2.5. In this scheme, the time stamp is generated using two counters. The least significant bit (LSB) counter, which is an N-bit modulo-M counter, toggles at every clock cycle. The most significant bit (MSB) counter is an M-bit binary-encoded counter toggling whenever the LSB counter rolls over. Because the MSB counter does not toggle at every clock cycle but every N clock cycles, it is serialized using the LSB counter as a bit-select and sent to pixels using one wire only. The LSB counter generates a global reference signal which stops all counting pixels. The values stored in pixels can then be decoded using a lookup-table (LUT). Note that if any arithmetic operations need to be done on-chip inside the pixel matrix with time stamps, the LUT approach is not feasible for even small values of k. This is true for Gray-encoded and pseudo-random time stamps as well. Later in this chapter, an arbitration technique using a time stamp comparison is shown which requires a comparison of two binary values.

This time stamping approach has a dead-time of M clock cycles. This

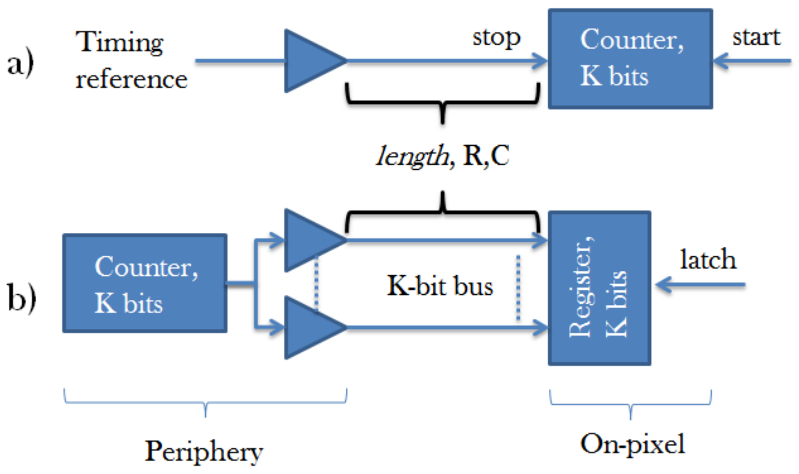


Figure 2.4: Two methods for measuring timing information in pixels.

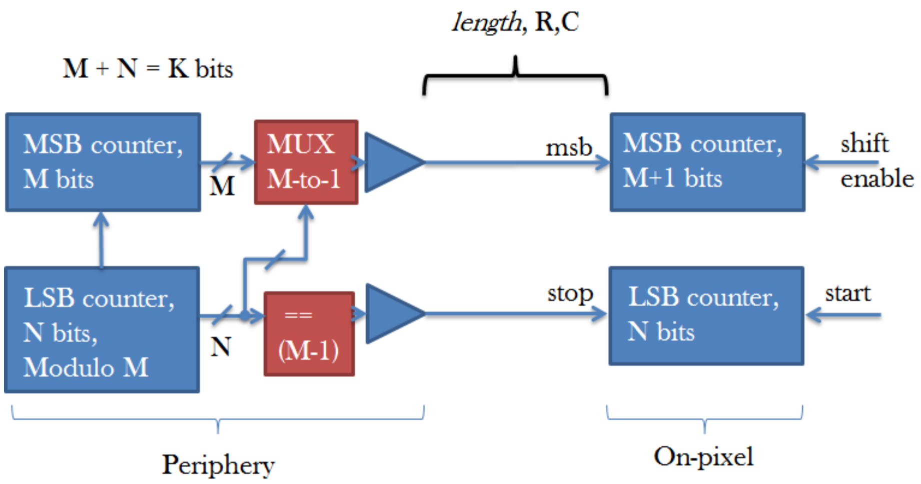


Figure 2.5: A novel method using serialization for generating a global timing reference for pixels.

is clearly better than exponential 2^K. It is not negligible as with a free running counter approach but the number of wires needed to distribute the time stamp is only two instead of K. This also means that only two drivers are required to drive the signals up the column. This approach also requires less signal transitions per clock cycle than Gray-encoded counter. It has been estimated from simulations that the average number of transitions per clock cycle is 0.59 for $K = 16$ and 0.7 for $K = 8$. As the K gets bigger, the global reference signal for the LSB counter switches less frequently, thus leading to even bigger reduction in the average number of transitions.

The time stamping presented in Section 2.3 is closely related to the concept of a bunch crossing. The Large Hadron Collider (LHC) at CERN uses a 40 MHz machine clock, and all detectors are synchronized to this clock. The time stamp in a detector in the LHC indicates from which bunch crossing data came from. Physically a bunch crossing is a collision of particles belonging to a specific bunch. The time stamp indicating the bunch crossing number is called bunch crossing identification data (BX-ID). Typically, when talking about the latency of data in a pixel chip, we are interested in the latency in relation to the time stamping range. Generally, the latency must be lower than the dynamic range of the time stamp to be able to unambiguously identify ToA information outside the readout chip.

2.4 Occupancy, hit rate and output rate

Occupancy is an indication of the utilization of the front-ends or an indication of the data traffic inside a chip within a certain period. It indicates which fraction of its resources a readout chip is using at a given moment. Occupancy of an HPD ASIC is defined as:

$$O = \frac{N_{pixels_{hit}}}{T\ N_{pixels_{chip}}} \times 100\% \tag{2.1}$$

where O is the occupancy, $N_{pixels_{hit}}$ the number of pixels containing hit information, $N_{pixels_{chip}}$ the number of pixels in a full chip and T is a period. In many applications, occupancy is a time-averaged quantity and the period is not explicitly mentioned. Occupancy has an impact on the choice of readout architecture as will be presented later in this chapter.

When a particle flux goes through an HPD, the detector is exposed to a certain hit rate. This flux can be constant or it can vary according to the environment in which the detector operates. The hit rate of an HPD ASICs is defined as:

$$R_{hit} = \frac{N_{pixels_{hit}}}{T_{acq}\ A_{chip}} \tag{2.2}$$

where R_{hit} is the hit rate, T_{acq} the acquisition time and A_{chip} area of a chip. Typically, hit rate is expressed as $hits/s/cm^2$. Intuitively, the output rate of a chip, R_{out}, must be at least equal to the input hit rate in order to avoid data losses. As shown later by simulations, this is not always a sufficient condition, as often the output rate must be at least slightly higher or data buffers large enough to decrease data losses.

2.5 Efficiency

In this thesis, only the efficiency of the digital readout architecture is analyzed. This means, for example, that inefficiencies of analog amplifiers are not taken into account. This cannot be ignored when calculating the efficiency of a full detector system but the analog front-end can be omitted when comparing the efficiency of different digital readout architectures and assuming the same analog front-end for all of them. The efficiency of a digital readout architecture is then defined as follows:

$$E_{ro} = \frac{N_{output}}{N_{input}} \tag{2.3}$$

where E_{ro} is the readout efficiency, N_{output} the number of correct hits received at the output of the chip and N_{input} the number of hits received from the output of the analog stage. E_{ro} must not be confused with tracking efficiency which is an attribute of a full detector system. Although not explicitly shown in (2.3), it must be emphasized that E_{ro} is always measured over a period. When counting the number of correct data packets transmitted from a chip, the latency of the packets must also be taken into account. Ambiguity in the time information can occur if the latency increases beyond the dynamic range of the timing information in the packet. For example, a packet with 14-bit time information recorded at 40 MHz can have a maximum latency of $(2^{14} - 1) \times 25\ ns\ = 409.6\ \mu$s before ambiguity in time stamping arises. As mentioned in Chapter 1, a quantization error can also be caused by the sampling of an analog signal with the digital clock. Before calculating E_{ro}, a maximum tolerable error must be defined. On the other hand, when comparing different readout architectures with each of them utilizing the same sampling or measuring technique, the quantization error can be ignored.

Figure 2.6 shows the readout efficiency of a single pixel as a ratio of $R_{readout}$ and R_{hit}. The 99 % threshold is shown with the red dashed line. The choice of this value is motivated later in this section. The plot has been obtained from simulation assuming exponentially distributed times-of-arrival for hits with a mean time of $1/R_{hit}$ between hits. It is also assumed that the pixel can buffer only one hit and discards all hits arriving between arrival of a hit and a readout operation. The readout operation is assumed to start after a new hit arrives into an empty pixel buffer. It can be seen from the Figure 2.6 that half of the hits are lost if $R_{readout}$ and R_{hit} are equal. Notice the improvement from 50 % to 90 % when increasing the readout rate by a factor of 10. Increasing the readout rate from 10 to 100 increases the efficiency by less than 10 %. The target of 99 % is reached when the ratio equals approximately 100. The efficiency of a single pixel E_{pixel} determines the maximum achievable E_{ro} of a full chip, when assuming perfectly uniformly distributed hits, because for a full chip, the E_{ro} can also be expressed as:

$$E_{ro} = \sum_{i=1}^{n} \frac{E_{pixel_i}}{n} \tag{2.4}$$

Note that (2.4) can be expanded to hold under any distribution of hits by weighing the E_{pixel} with a number of hits in that pixel against the total number of hits for the full chip:

$$E_{ro} = \sum_{i=1}^{n} \frac{E_{pixel_i} \ \ hits_i}{\sum\limits_{j=1}^{n} hits_j} \tag{2.5}$$

where $hits_i$ and $hits_j$ indicate the number of hits in a pixel i or j.

The minimum efficiency required is solely determined by the application. For example, in [49], an efficiency of 97 % is quoted. [37] targets an efficiency above 99 %. In [56] an efficiency of 98 % is deemed acceptable for efficient track reconstruction. Efficiency of 99 % is targeted In [57]. In this thesis, a minimum required efficiency of 99 % is used as a target. Because most of the data analysis in pixel detectors is performed outside the readout ASIC, factors related to external electronics also play a role in determining the final required efficiency for a readout ASIC.

2.6 Continuous and sequential readout

Figure 2.7 illustrates the concept of sequential readout. When a chip is in a data acquisition mode, it is accepting hits at its inputs for further processing. These hits are shown as green arrows in Figure 2.7. In a sequential architecture, the data acquisition can be disabled (T_{off}), and incoming signals from the sensor are not fully processed by the readout chip (red arrows). Signals from the sensor may still be processed by the analog electronics but no digitization of this data is done.

Acquisition time T_{acq} indicates the duration of the data acquisition. Off-time T_{off} indicates how long an architecture is insensitive to events after

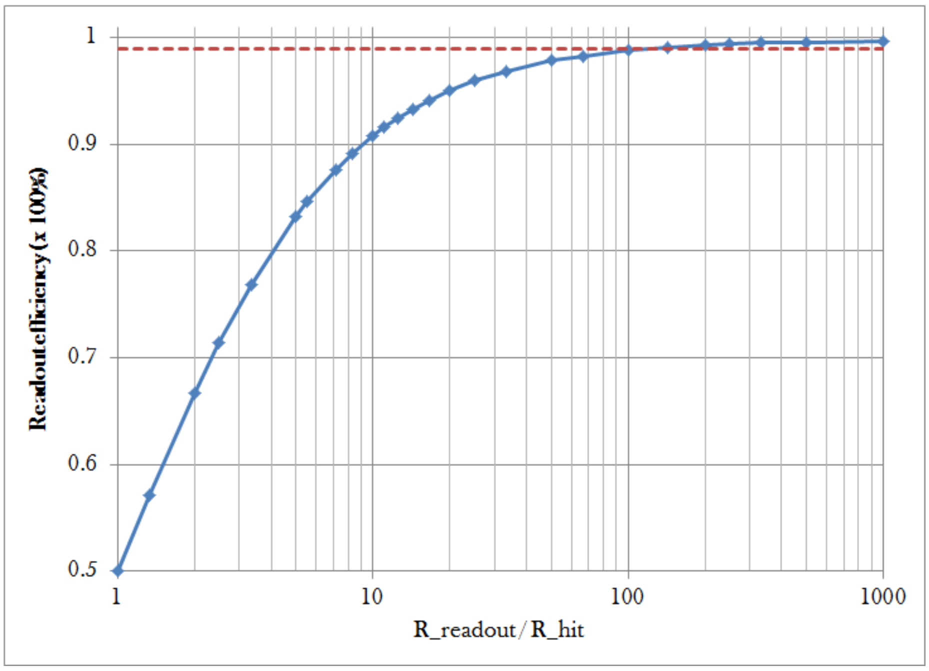


Figure 2.6: Readout efficiency of a single pixel with different readout and hit ratios.

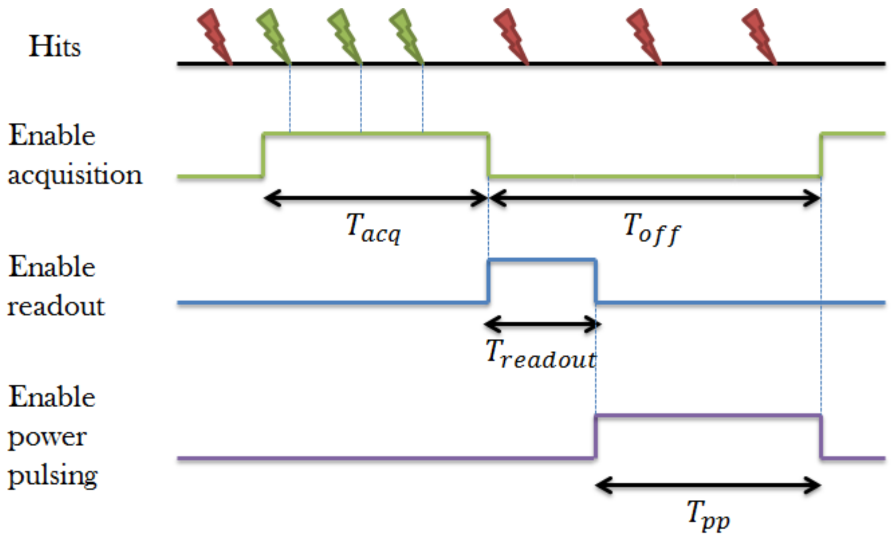


Figure 2.7: Acquisition, readout and power pulsing phases of the operation of an HPD readout ASIC.

acquisition. T_{off} may be divided into multiple phases such as readout ($T_{readout}$) and power pulsing (T_{pp}) phases, discussed below. A duty cycle of the chip can be defined using acquisition- and off-times as follows:

$$T_{duty} = \frac{T_{acq}}{T_{acq} + T_{off}} \times 100\% \tag{2.6}$$

where T_{duty} is the duty cycle of the chip. For example, for 1 ms acquisition time with 1 ms of off-time, the duty cycle is 50%. In theory, a continuous readout is defined to have a duty cycle of 100% and $T_{off} = 0$ but in reality for a continuous readout $T_{off} << T_{acq}$ and $T_{off} > 0$. In some applications, T_{off} can be significantly longer than $T_{readout}$. In such cases, the power consumption of the chip can be reduced by powering down some parts of the chip for the remainder of T_{off}. This functionality is called power-pulsing or duty-cycling. This is in general only beneficial if the application allows a small T_{duty}. So power-pulsing is not applicable as a generic power reduction technique, like clock-gating for example. Architectures presented in this thesis are optimized for very low T_{off}, and are designed for applications requiring T_{duty} close to 100%.

2.7 Full and zero suppressed readout

Each pixel in an HPD can collect up to b_{pixel} bits of information. Therefore, for a full chip the maximum amount of bits can be defined as:

$$N_{bits} = b_{pixel}\ N_{pixels_{chip}} \tag{2.7}$$

where N_{bits} is the total number of bits to be read out. A pixel need not be the smallest unit of readout on a chip but (2.7) holds even in the case of larger unit (for example a super pixel). It can be said that a readout is zero suppressed if $N_{bits_{readout}} < N_{bits}$ for some $N_{pixels_{readout}} < N_{pixels_{chip}}$. Zero suppression always requires extra information to be added into the data, so above a certain value of $N_{pixels_{readout}}$ in a zero suppressed architecture, the $N_{bits_{readout}}$ becomes larger than the N_{bits} would be for a full unsuppressed readout. Instead of using the term *zero suppression*, a term *data encoding* can be used as well. Zero suppression can also be thought of as a form lossless compression, and can be applied to any bitmap.

2.8 Readout and acquisition control with a shutter

One significant benefit of active pixel sensors over passive ones like CCDs is that an electronic shutter can be substituted for a mechanical shutter. An electronic shutter can be precisely controlled inside an ASIC which makes it possible to open and close the shutter in few nanoseconds. This allows for precise control of measurement and off-times. For example, if a chip cannot cope with the hit rate it is exposed to, the shutter can be closed to give the chip time to recover. As will be shown later, by precisely controlling the shutter, for each ratio of R_{out} and R_{hit}, E_{ro} can be improved by selecting a specific T_{acq}.

2.9 Triggered and trigger-less architecture

A trigger signal can be used to select an event of interest inside a chip. In general this means the amount of data produced by a readout chip can be reduced by filtering with the trigger signal. The signal can be either external to the system or generated internally on-chip.

In large-scale applications like high-energy physics experiments, dedicated detectors generate the trigger signals after some latency for processing. These triggers are then transmitted to the other detector systems. Triggers are usually divided into several levels with the lowest level trigger having the lowest latency. Its latency can be several microseconds, for example in [58], but it is determined solely by the application. This requires more buffering inside the readout chip than in a trigger-less approach as data cannot be transmitted until the trigger arrives.

Reduction of data by using a trigger is not defined as zero suppression as specified earlier. Architectures operating without a trigger cannot discriminate between events of interest using time stamps and thus need to transmit all data off the chip. The focus of this thesis is on trigger-less readout architectures only.

2.10 Readout modes

Readout architectures can be broadly divided into three categories: frame-based, packet-based and hybrids of these architectures. Each of these solutions can be a full chip readout or zero suppressed as presented in Section 2.7. In terms of hardware costs, the frame-based architecture has typically the lowest cost. This is due to the fact that the same memory elements inside pixels that are used in the data acquisition can be re-used as a shift register for shifting the acquired data down.

2.10.1 Full-frame and packet-based modes

Data encoding for frame-based and packet-based architectures are shown in Figure 2.8. There is no general definition of either mode so the following categorization is assumed here. In a frame-based architecture in Figure 2.8 a): To decode the full position of hits, a structure of distributed address

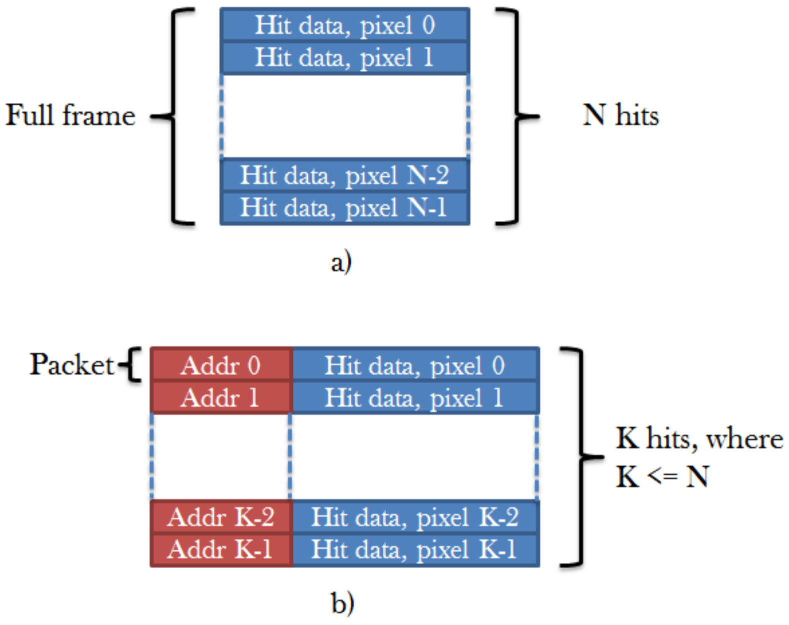


Figure 2.8: Frame- and packet-based data structures for encoding pixel hits.

bits must be assembled off-line and actual hit addresses decoded from this information. Therefore, a logical structure of a full chip must operate to encode all address information. Note that in the case of a full, non-zero suppressed frame, pixel addresses are implicitly encoded into the position of data bits in the frame.

In a packet-based architecture in Figure 2.8 b), the address information of a pixel is directly encoded in a data structure transmitted off the chip and can be directly read from the data-flow. In general this means that there exists a logical readout structure inside a chip which is smaller than the full chip itself. Note also that a chip utilising a packet-based architecture sends out its information only if a pixel corresponding to that packet address has been hit.

2.10.2 Zero-suppressed frame and hybrid mode

A zero-suppressed frame is shown in Figure 2.9. The structure is a frame-based according to the previous definition because each pixel on a chip sends its status bit in the structure. There is no way to know a location of a specific pixel unless a full set of status bits is analyzed, and addresses decoded from this frame. However, it is not considered as a full frame because each pixel sends its hit data only in a case where its status bit is 1.

A hybrid solution is to send the data off the chip in a data structure based

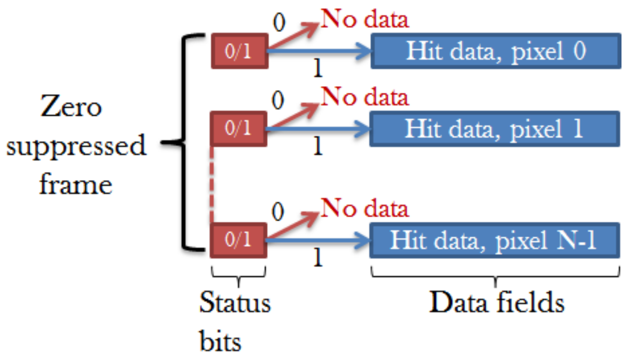


Figure 2.9: Zero-suppressed frame structure for encoding pixel hits.

on sub-sections of the matrix and incorporating hits from several pixels. The number of sub-sections must be less than the number of pixels in a full chip, otherwise the solution in question is a frame-based architecture. It must also be larger than a single pixel to avoid falling into a packed-based architecture. A hybrid solution between a frame-based and a packet-based is shown in Figure 2.10. Note that the k shown in the figure can be an arbitrary positive integer, and need not be the same for all data structures. Figure 2.10a shows a structure with one address and several hit data from pixels. The address of hit data is implicitly encoded into its position in the data structure. The address must identify the position of the sub-section within the pixel matrix. Note that every pixel in the sub-section must send data bits, even if they do not contain any information. Figure 2.10b shows a structure where addresses of hit data are explicitly shown in the structure. This structure is especially useful if the packet has a fixed length and always contains a constant number of pixel data but the data can come from different addresses. Information from some pixels can also be removed from the structure if an additional field indicating the number of hit data is present (Figure 2.10c). As the fourth technique, a bitmap of pixels can be included in the packet to indicate a valid data in the corresponding pixel (Figure 2.10d). As these structures can also be considered as packets, from now they are referred to as super pixel packets (SPPs) and the sub-sections of the pixel matrix as super-pixels. There are also another options for encoding the data such as used in [37] where a time stamp is shared between several pixel data.

As the last architecture, a generic zero suppression scheme of a pixel matrix is shown in Figure 2.11. There are x columns and y rows in the matrix, and it has been divided into distinct regions and sub-regions of pixels. At the lowest region, the number of pixels is h_0v_0. At the highest level, the size of a region is a product of all region sizes: $v_0h_0...v_ih_j$. Note

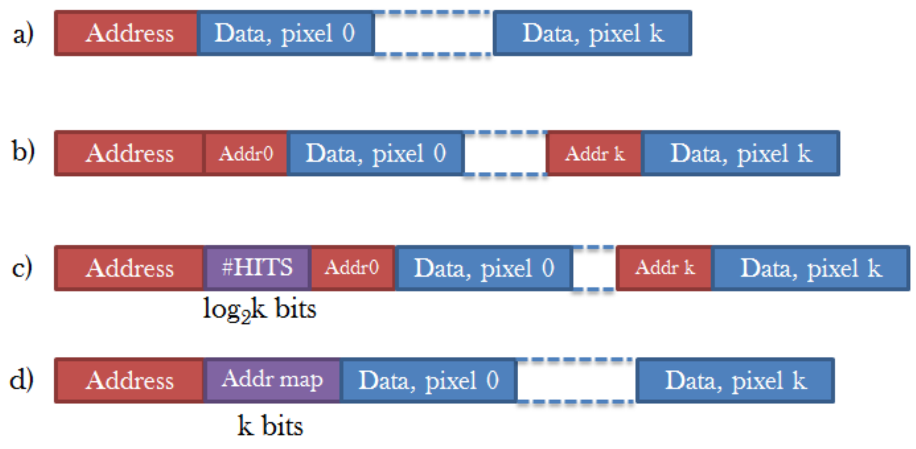


Figure 2.10: A hybrid data structure for encoding pixel hits.

that here $i = j,\ i, j \in \mathbb{N}_0$.

Figure 2.12 shows a similar approach for zero suppression carried out at the periphery level. The smallest blocks map directly to the largest width of a pixel region. These top-most blocks are at the lowest level of periphery zero suppression. If the largest pixel region is 4 pixels wide ($h_0 ... h_j = 4$), each top-most block in Figure 2.12 represents these 4 pixel columns. If none of these columns have at least one hit, a flag bit in this block is set to 0, otherwise it is set to 1. Ultimately at the lowest suppression level in the periphery, it is possible to have 1 flag bit indicating if the chip has any data at all.

To distinguish different zero suppressed architectures from each other in the following discussion, three different notations are defined:

$$Vertical\, region\, division:\ V(v_0, v_1, ..., v_i) = \{v_0, ..., v_i\} \tag{2.8}$$

$$Horizontal\, region\, division:\ H(h_0, h_1, ..., h_j) = \{h_0, ..., h_j\} \tag{2.9}$$

$$Periphery\, region\, division:\ P(p_0, p_1, ..., p_k) = \{p_0, ..., p_k\} \tag{2.10}$$

where $i, j, k \in \mathbb{N}_0$. These notations simply encode the size of the different pixels regions. They are not functions and they do not map to any scalar but simply list the sizes of different regions as a list of integers. To give an analogy to an existing design, for the readout architecture in [35] we can use notation $V(8), H(2)$ as this readout architecture utilizes a 2×8 super

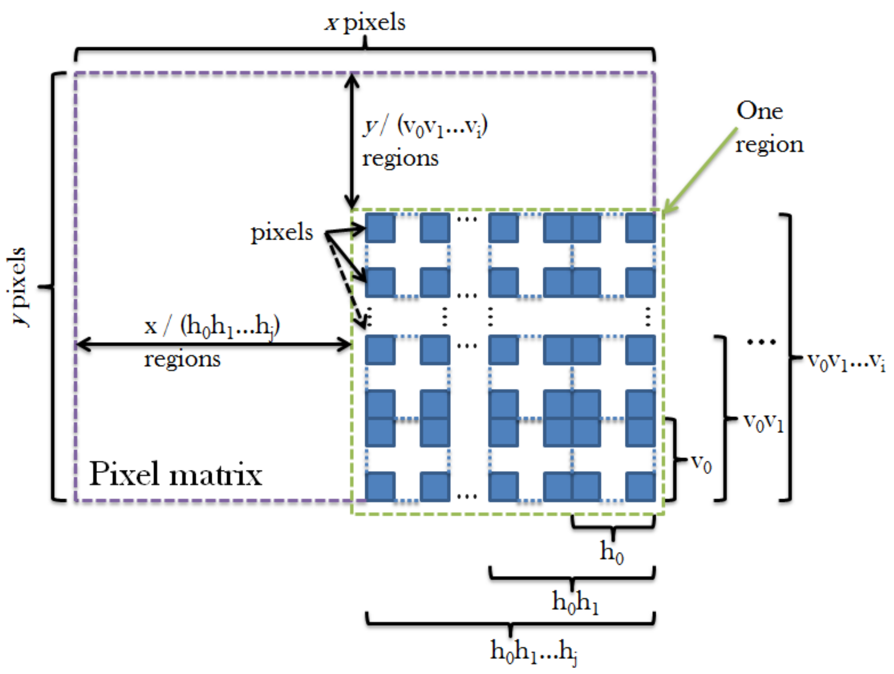


Figure 2.11: A zero suppression scheme for a single pixel frame.

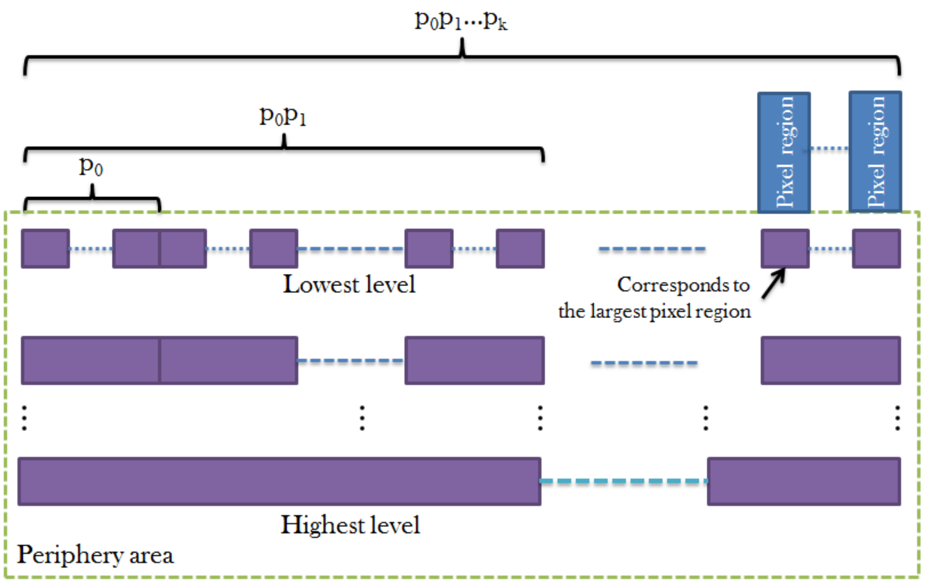


Figure 2.12: A zero suppression scheme for a chip periphery.

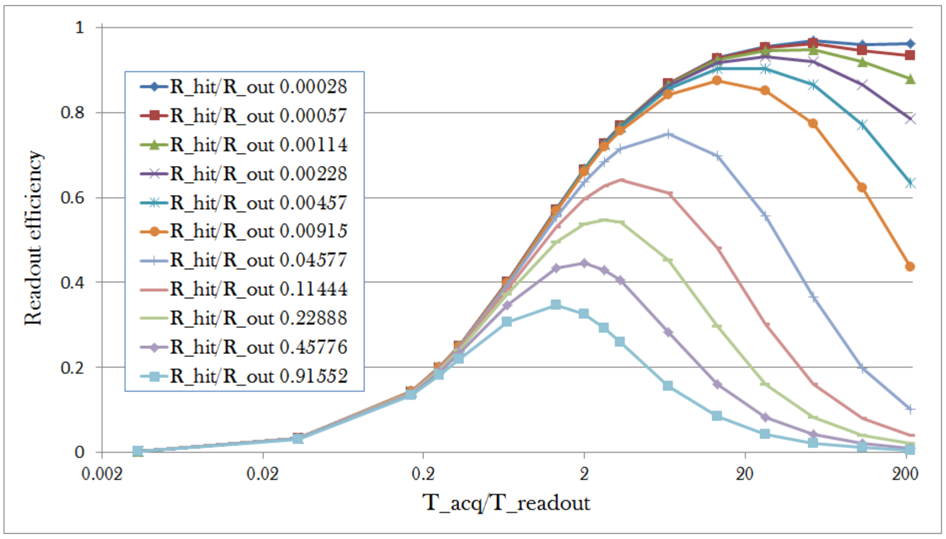


Figure 2.13: Readout efficiency of frame-based readout with different acquisition times.

pixel structure for zero suppressing the data. In that architecture, two pixel columns are mapped into one flag bit at the periphery level. Because there is no further suppression of these flag bits, there is no zero suppression at the periphery level.

2.10.3 Analysis of frame-based readout mode

Packet-based and hybrid readout architectures in general introduce complexity in the pixel circuitry, so the disadvantages of frame-based architectures must be fully assessed before making any architectural decisions. Figure 2.13 shows the readout efficiency of a full-frame readout architecture with different acquisition times T_{acq}, simulated with different ratios of output and input hit rates. In this case, it is assumed that N_{pixels} is $256 \times 256 = 65536$. It can be seen that when the ratio of R_{hit} and R_{out} approaches unity, the maximum achievable E_{ro} drops below 40 %. It can be concluded from Figure 2.13 that to achieve E_{ro} high than 90 % for example, the R_{out} must be 200-fold compared to R_{hit}, and $\frac{T_{acq}}{T_{readout}}$ must be chosen accordingly. This ratio can of course be chosen freely if an electronic shutter is implemented inside a chip. In fact, software used in [59] to control a Timepix chip [25] implements an online analysis algorithm which adjusts the shutter length to the rate of detected particles for maximum readout efficiency.

2.11 Comparison of zero suppression schemes

All packet-based architectures and hybrid solutions fall into the category of zero suppressed architectures. As discussed in Section 2.7, there exists a break even point at some occupancy below which the total data volume read out using a zero suppressed architecture is less than the data volume read out using a full readout.

Figure 2.14 shows the total number of bits produced with different zero suppression schemes at different pixel occupancies. A full frame readout is included for comparison. 32 bits per pixel hit are assumed with a pixel matrix of 256×256 pixels. The physical dimensions of a pixel are irrelevant for this comparison. Based on the definition in Section 2.10 and Figure 2.12, the $V(1), H(1)$ architecture skips all columns with no hits and reads all status bits from columns with at least one hit. Each pixel contains a status bit and the full 32 bits of a pixel are read out only if a pixel status bit is 1.

$V(8), H(2)$ in Figure 2.14 indicates that only one bit per a 2×8 region of pixels is sent off the chip if none of the pixels in that region are hit. If there is at least one hit, all 16 pixels will send their status bits but only pixels with status bit set to 1 send the full 32 bits. With $V(8,8), H(2,1)$ scheme, pixels are combined into 2×8 regions first, then 8 of these regions are vertically combined into one super region. Now there are $\frac{256}{8 \times 8} = 4$ status bits always sent from each column having at least one pixel hit.

$P(4)$ in Figure 2.14 indicates that EoC regions are divided into groups of 4 and only 1 status bit per group is sent off the chip if none of EoC regions in that group are hit. For Figure 2.14, a break even point between full-frame and packet-based architecture exists with occupancies $> 70\%$. For zero-suppressed frame architectures this break even point is above 90%. It should be noted that for occupancies very close to 100%, a full-frame readout produces always the least amount of data due to the absence of any kind of address or status bit information.

The $V(8), H(2)$ scheme has been implemented in [35] using 65 nm CMOS technology. In [25, 47] the full-frame readout is implemented. Packet- or SPP-based architectures are presented in [37, 45, 49, 54]. It is also reported in [36] that implementing $V(8), H(2)$ scheme gives a reduction of 25% of data rate when assuming 2×2 pixel clusters.

2.12 Data buffering

2.12.1 Front-end efficiency and data buffering

Consider a system shown in Figure 2.15 having one data buffer with depth $N > 0$. This is representative of a buffer in a single pixel. Two processes, P1 and P2, operate on this buffer. P1 writes data into the buffer at rate

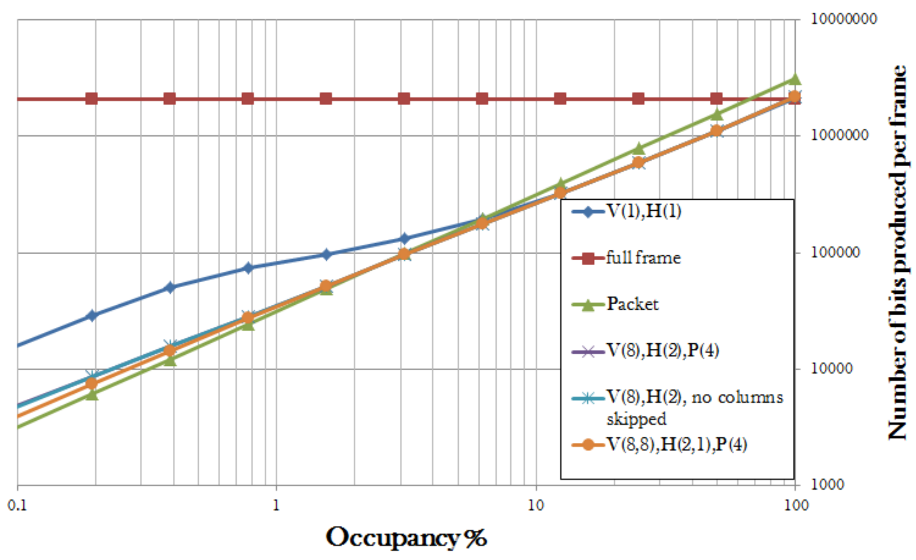


Figure 2.14: Total number of bits versus occupancy.

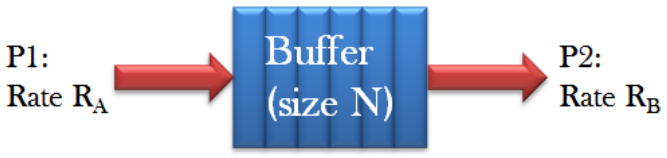


Figure 2.15: A buffered system depicting a digital pixel front-end electronics.

R_A and P2 reads data from the buffer at rate R_B. It can be said that for any finite N, if $R_A > R_B$, an overflow will occur eventually. If $R_B \geq R_A$, and rates are constants assuming also no burst-writes, for any N $>$ 0, no possibility of an overflow exists. Note that in the model used here, data is removed from the buffer after the fixed service period $\frac{1}{R_B}$ has passed.

In many pixel applications the production of data is governed by Poisson statistics and the rate A is not a constant but an average rate of occurrence of writes into the buffer in this case [60]. In fact, the system in Figure 2.15 is an instance of $M/D/1/N$-queue [61], where M indicates Markov process, D indicates deterministic service time, 1 indicates the number of servers and N indicates the size of the buffer. Figure 2.16 shows the simulated efficiency as a function of ratio of R_B and R_A with different N. This plot indicates which buffer size N and readout rate R_B must be chosen for a rate R_A so that a specific efficiency can be achieved. When the buffer is able to hold only one hit in memory ($N = 1$), a ratio $\frac{R_B}{R_A}$ must be $>$ 100. When increasing the depth of the buffer from 1 to 2 slots, a reduction of 20 in the R_B can

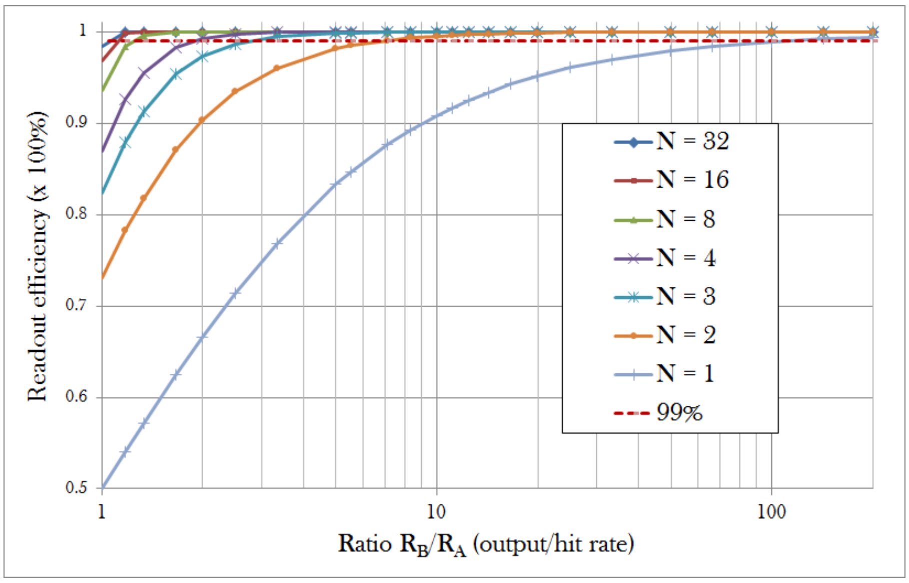


Figure 2.16: Readout efficiency of a digital front-end buffer as a function of the buffer depth and the ratio of readout and hit rates.

be achieved for $E_{ro} > 99\%$. For an increase of the depth from 2 to 3, a reduction of less than 3 is achieved.

A digital memory buffer with an overflow control is essentially a non-paralyzable system [60]. An analytical formula for the efficiency of this kind of system is [60]:

$$n = \frac{m}{1 - m\tau} \tag{2.11}$$

where n is the true rate, m is the recorded rate and τ is the dead time of the system. By substituting $E_{ro} = \frac{m}{n}$, we obtain the function for the dead time:

$$\tau = \frac{1 - E_{ro}}{E_{ro}\ n} \tag{2.12}$$

This result is intuitive, as we decrease the requirements for E_{ro}, τ can be increased for the same true input rate n. Also, for the same E_{ro}, if n is decreased, the requirements for τ are also relaxed. The Equation 2.12 also confirms the results obtained for N = 1 in Figure 2.16. This is an important validation as the buffer model will be used as a basic building block for larger systems in the following parts of this thesis.

2.12.2 Pixel column readout

Consider a system shown in Figure 2.17. In this system, k data buffers are connected to a data fabric. The fabric can be a bus-based system or a

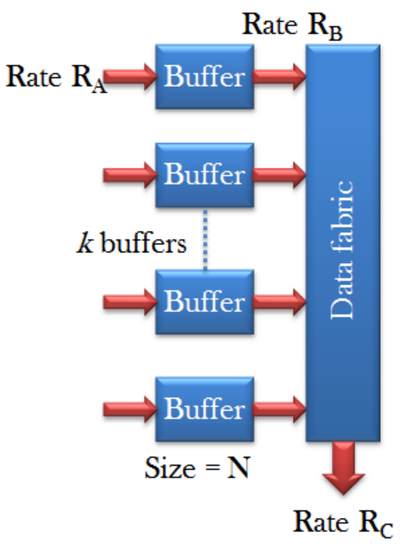


Figure 2.17: A system with pixel buffers connected to a data transportation fabric.

more complex network of switches, data buffers and registers. This system can also be thought of as a $k \times 1$ crossbar where k inputs are mapped to one output. This is representative of a complete pixel column and its data transmission scheme. If it is assumed that $k \times R_B = R_C$, each buffer has equal priority to the data fabric. This equal priority can be implemented using a token ring arbitration. In this scheme each buffer contains a flip-flop, and the token travels unidirectionally through the flip-flops. However, this is inefficient allocation of bandwidth as a buffer may not have data during each of its time slots while another buffer might need more than 1 time slot. Also, based on the analysis in the previous section, for $N = 1, 2$, $\frac{R_B}{R_A} = 100, 7$ respectively for $k = 1$. This implies a need to "over design" the rate R_C if readout efficiencies above 99 % are desired. For example, assuming $N = 1$, $k = 1$, R_C must be equal to $R_b \geq 100R_A$. It must be noted that the requirement for this ratio is lowered when k is increased.

The effect of increasing k, the total number of buffers in a column, is shown in Figure 2.18. It is assumed that $N = 1$. It can be seen that the higher the k, the lower the ratio of R_C to R_A must be. Fair arbitration and equal time multiplexing of bandwidth allocation is assumed here. In terms of designing larger chips with more pixels per column, Figure 2.18 clearly shows that the higher number of pixels per column is beneficial in terms of the readout efficiency. On the other hand having a higher number of pixels per column introduces other complications, for example longer signal delays from the top of the column to the bottom, and increases the power

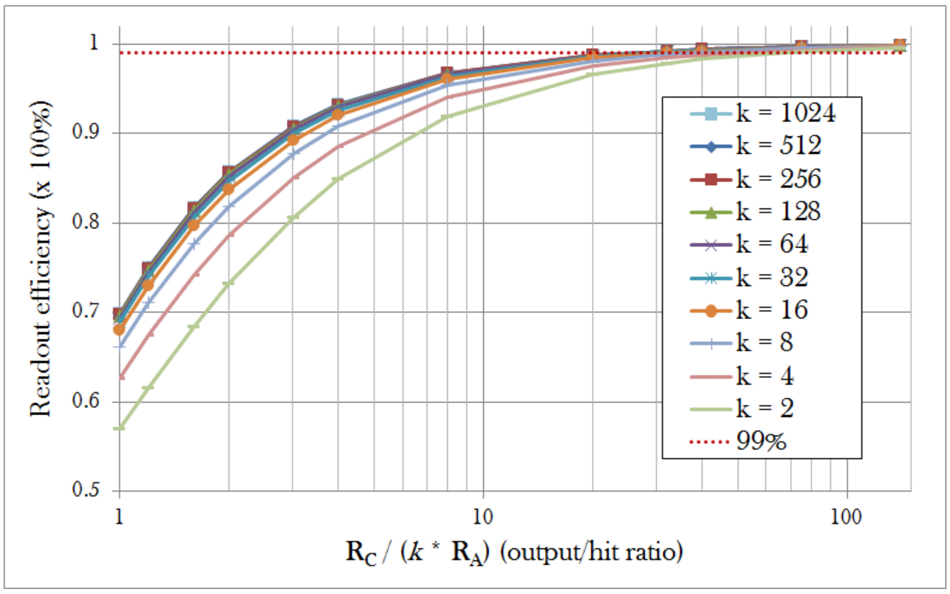


Figure 2.18: Readout efficiency of a pixel column with different number of buffers per column.

consumption per column.

As mentioned before, it is not particularly efficient to allocate a fixed time slot for each of the data buffers. An ideal arbitration scheme would allocate a time slot instantly to a buffer, and would read the buffers in a strict first-in first-out (FIFO) order respect to the arrival times of hits. This claim is not proven analytically here but only shown by a simulation. Two different arbitration schemes are shown in Figure 2.19. In the Figure 2.19a, a FIFO arbiter is shown. Requests from buffers are processed in FIFO order to minimize the waiting times of buffers. In the case of simultaneous requests, it is assumed that all requests are placed randomly into the arbitration FIFO. In the Figure 2.19b, another arbitration scheme using a token ring arbitration is shown. The scheme consists of two levels of token rings, and is a special case of a tree of arbiters constrained to two levels. Variable G denotes the number of token arbiters in the inner ring. A tree arbiter has been presented in [62] for a 128×128 pixel image sensor. A 7-stage arbiter is created as a binary tree using small arbiter elements, and used to select a column and a row among active requests.

A token circulates in the outer ring until it finds an inner ring with an active request. In both the cases, it is assumed that $T_{arbitration} + T_{data} = \frac{1}{R_C}$. T_{data} is the time taken to send data from a buffer to the receiver. To be more specific, in the case of a token ring, traversing one hop in the outer ring takes $T_{arbitration}$ and traversing one hop inside the inner ring takes the

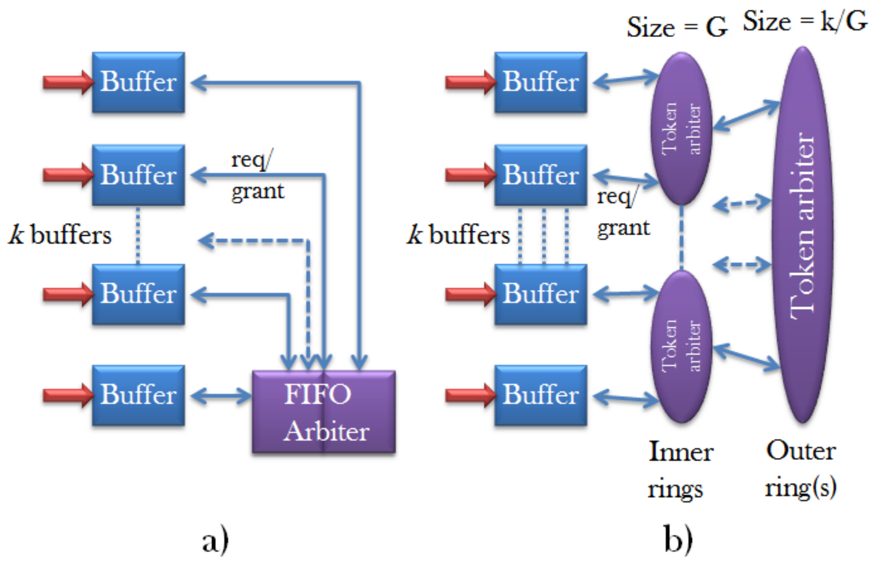


Figure 2.19: Two column arbitration schemes: a) FIFO arbitration b) Token ring arbitration.

same amount of time. But the inner ring is only entered if and only if there is at least one active request in that ring.

The impact of different arbitration schemes is shown in Figure 2.20. It is assumed $N = 1, k = 256$ and exponentially distributed times-of-arrival as before. As expected, when $G = 1$, E_{ro} is the lowest due to the highest arbitration delay. In this case, there is effectively no outer ring in the arbiter as each hop always takes $T_{arbitration}$ regardless of the state of the corresponding request-signal. When changing G from 1 to 4, the 99 % E_{ro} is achievable with 4 times lower R_C. A two-fold increase of G from 4 to 8 still gives a reduction of 2 for R_C but beyond that the gains are less significant. It should be mentioned that higher values of G require signals to be driven for longer distances. This is taken into account in the circuit-level analysis in the following chapters. A study in [39] has implemented a token ring scheme with $G = 64$ for a column of 256 pixels with a pitch of 55μm. However, no physical implementation details are shown nor post-layout analysis is presented.

As the last option, a priority-encoded scheme may be used as presented in [53, 63]. This scheme always selects the first pixel or super pixel having data at the bottom [63] or at the top [53] of the column, and thus it is not a fair arbitration scheme when using a continuous data taking mode. It works well in a frame-based readout mode because the number of hits per pixel is limited by the duration of the shutter and the readout mode, but in the

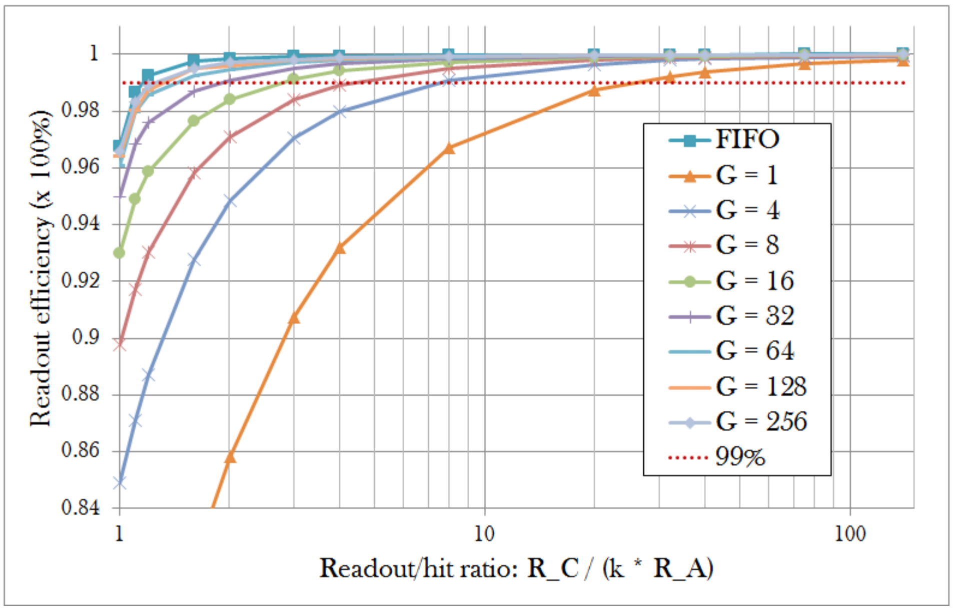


Figure 2.20: Readout efficiency of a pixel column with different arbiters.

continuous mode this possibility does not exist. This option is attractive for low-power applications though, because the digital power consumption $< 10mW/cm^2$ is reported in [63].

2.12.3 Super pixel buffering

From Figure 2.16 it can be seen that two means for improving efficiency are increasing the buffer depth or increasing the ratio of readout to hit rate. As an increase in the buffer depth can double the required memory per pixel, and an increase in the readout rate will require faster clock speeds thus consuming more power, another solution is examined. Consider a system in Figure 2.21 with m pixels and one super pixel buffer. The minimum required rate R_C is then $m \times R_A$. Unless the rate $R_B > R_C$ at least instantaneously, using a second stage buffer larger than $M = 1$ serves no purpose as the buffer will always contain one hit at most. In the case of Poisson arrivals, the instantaneous rate for R_B may be higher than R_C although the average rate of R_B would be $< R_C$.

Figure 2.22 shows the efficiency of the super pixel system presented in Figure 2.21. The results have been obtained fixing the depth of the buffers in the first stage to N = 1, and changing the depth of the second buffer (M). It is also assumed that $\frac{R_B}{m \times R_A} = 100$. The mean hit rate is held constant and the rate R_C is changed. The variable $m = 8$ in this case, and a token ring arbitration is assumed, where $G = m$. As shown in Figure 2.20, with $G = 8$

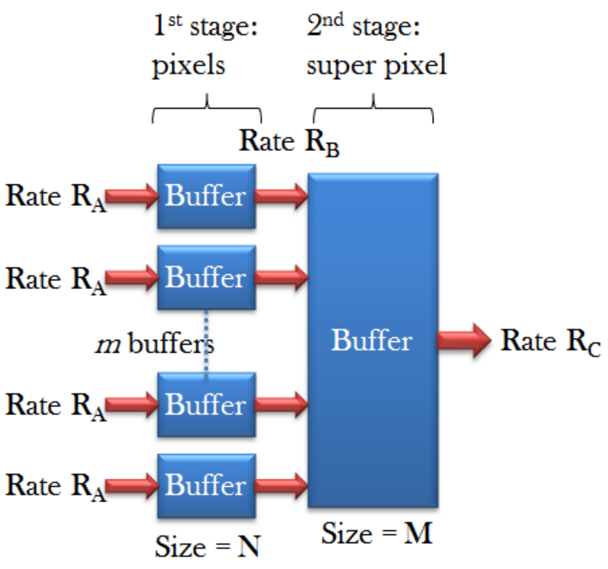


Figure 2.21: A buffered system with m pixels and one super pixel buffer.

the rate $\frac{R_C}{k \times R_A}$ must be 5 times higher than with ideal FIFO arbitration.

Compared to the case of a single pixel and Figure 2.16, it can be seen that even when M = 1, the 99 % E_{ro} is achieved using a $\frac{R_C}{m \times R_A}$ of less than 4. This result shows that with a shared buffer of one extra slot, a reduction of the readout rate from 100 to less than 4 can be achieved without introducing any losses to the readout efficiency. It is true that R_B must still be chosen according to the previous analysis based on Figure 2.16. However, when analysed later in this thesis, it becomes clear that R_B must only be a local rate to the super pixel, while R_C has to be a rate at the column level. When comparing a single pixel case where the buffer size $N = 2$ to a super pixel case with $N = 1, M = 1, m = 8$, the additional buffer slot is effectively shared amongst 8 pixels. Assuming that buffers are of equal size and type (one buffer slot = 1), and adding an extra overhead of one buffer slot to the super pixel case, the area reduction can be expected to be

$$\frac{8 \times \ \ single\, pixel, N = 2}{super\, pixel, N = 1, M = 1, m = 8} = \frac{8 \times 2}{8 \times 1 + 1 + 1} = 1.6 \qquad (2.13)$$

This is only a coarse analysis of the hardware resource required, and this claim will be more accurately substantiated when the circuit-level analysis is performed in the following chapters. Using a buffer structure shown in Figure 2.21 helps to achieve a high efficiency whilst reducing the clock frequency and the required driving strength for repeaters which drive the signals distances of over 1 cm up the pixel column.

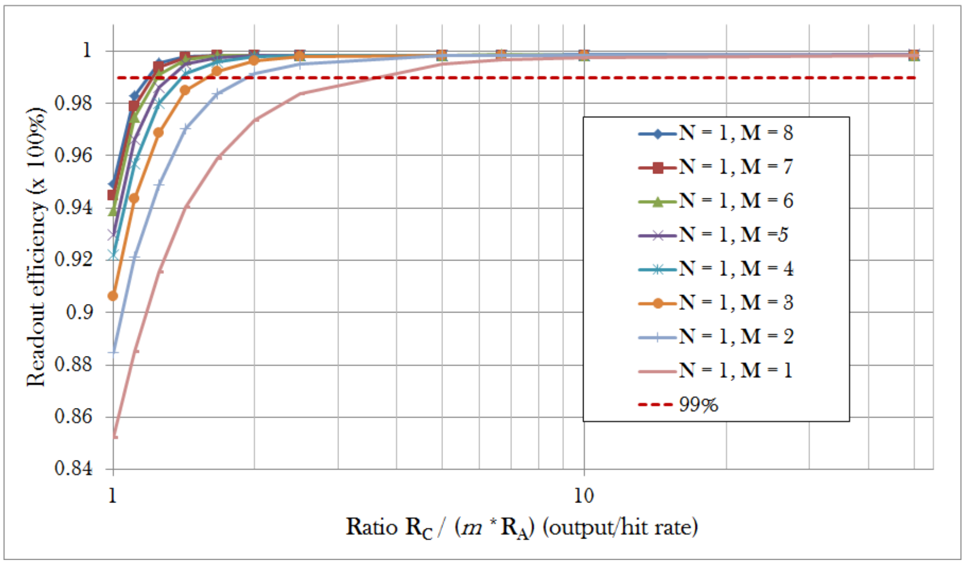


Figure 2.22: Readout efficiencies with super pixel buffer depths $M = 1...8$ and different readout to hit rate ratios.

After having established the means to decrease readout rate R_C by introducing an intermediate buffering stage, a larger system using this intermediate buffering is analysed. The system under analysis is shown in Figure 2.23. The full system has k pixels or buffers at the first stage and $\frac{k}{m}$ super pixels or buffers of the second stage, where m is the number of pixels connected to one super pixel buffer. Each of these buffers is connected to a data fabric which represents a bus or any other network suitable for transportation of data.

2.13 Node-based data fabrics

2.13.1 Linear node-based data fabric

In this section, a novel architecture using a node-based approach is presented. In essence, this is a variation of a "conveyor belt" scheme mentioned in [12], but with node-based arbitration and without having to implicitly embed the time stamping into the readout scheme. It means there is no global controller initiating the data transfer between the nodes. The scheme presented here is entirely data-driven, meaning that the nodes will start the data transfer only if there is valid data to transmit. This also means that if there are nodes with higher input data rate, they can send this data rapidly to the next node if that node has much lower input rate.

For this novel architecture, consider a network of buffers and data nodes

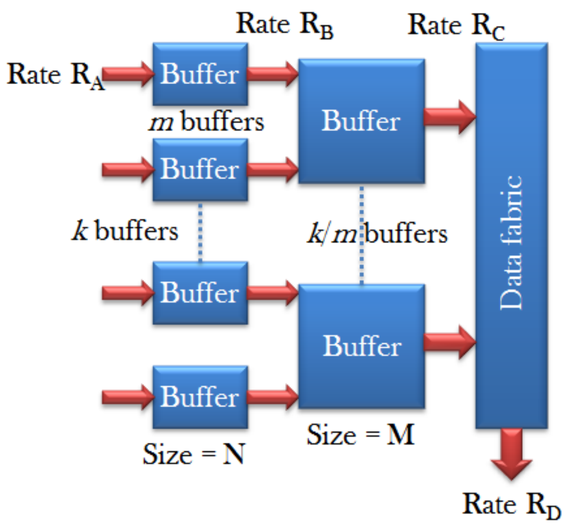


Figure 2.23: A buffered system with k pixels, m pixels per a super pixel and k/m super pixels connected a data fabric.

shown in Figure 2.24. This network can also be seen as a hardware queue with an insert-capability in each node. The node stores its state and the data locally, and there is no global controller for the full queue. In this scheme, data propagate from the top to the bottom, and each buffer has to send its data through a number of nodes. There is no direct connection from the top to the bottom in a bus-like manner, and several data can move between different nodes simultaneously. The fabric differs from a typical priority-queue hardware implementations [64] because it is used for data transport, and not explicitly for sorting the data. An important assumption made here is that packets passing through nodes are of equal length. In fact, each node can be considered a specialized version of a register in a shift-register based queue presented in [65]. An important distinction is that in Figure 2.24 each node has two inputs instead of only one. This indicates that an internal arbitration algorithm must be implemented.

This system also has similarities to pipelined bus structures [66, 67] which can reach rates of Gpackets/s between two nodes in 180 nm technology. The major difference is that the data fabric is a simplified version of these because it is unidirectional, and each node will not transmit data back to the local buffer but will only receive data from it. This will limit the rate achievable for data fabric to R_{C0} as can be concluded from Figure 2.24 because each data packet must eventually pass through the last node in the chain.

Again assuming a uniform input rate, R_A for each data buffer, the minimum requirements for the rates R_{Ci} are different for each $i = \{0, 1, \cdots, k -$

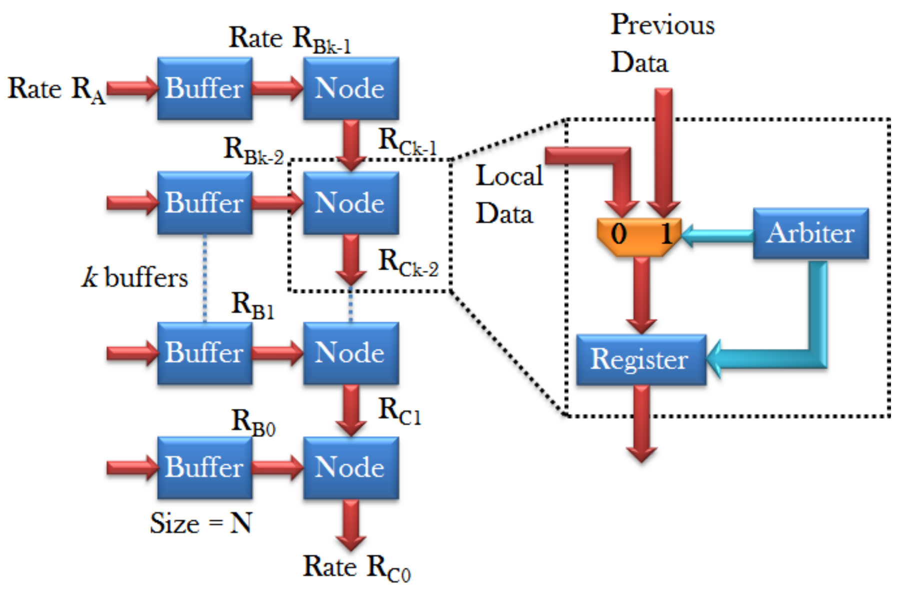


Figure 2.24: A buffered data fabric with k buffers (pixels), and k data nodes.

1}, and are defined as follows:

$$\begin{aligned} R_{Ck-1} &\geq R_A \\ R_{Ck-2} &\geq 2R_A \\ &\vdots \\ R_{C1} &\geq (k-1)R_A \\ R_{C0} &\geq kR_A \end{aligned} \tag{2.14}$$

Each R_{Bi} is subject to the same restrictions as presented before. If $N = 1$, the ratio $\frac{R_B}{R_A}$ must be greater than 100 for the readout efficiency $E_{ro} \geq 99\%$. A uniformly random distribution for buffer locations and an exponential distribution for the times-of-arrival are assumed.

Each buffer has to gain access to its local node before it can transmit any data. The arbitration between the local buffer and the previous buffer is also shown in Figure 2.24. If an equal priority is given to the local and previous buffer, an unfair arbitration is utilized. This has also been called a *parking lot problem* [68]. The buffer at the bottom of the column has the highest priority of $\frac{1}{2}$, the second buffer $\frac{1}{4}$ and finally the *nth* buffer a priority of $\frac{1}{2^n}$. The topology in Figure 2.24 cannot have a perfectly fair arbitration due to its asymmetrical structure. However, there are approximation algorithms which perform approximately fairly locally in each node. Based on an algorithm, the arbiter selects data either from the local buffer or from the previous buffer. This algorithm is only used in the case of conflicting requests from

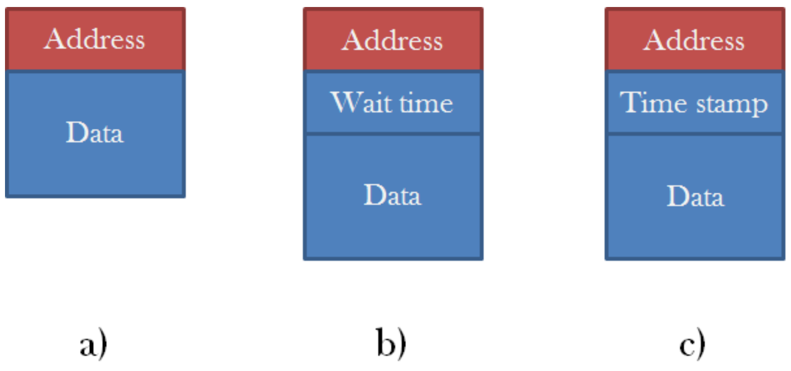


Figure 2.25: The contents of a data packet for different arbitration schemes: a) WRR, b) LWF and c) OCF.

the two buffers. Different algorithms used in the simulations are:

1. weighted round-robin (WRR)
2. Longest-Wait-First (LWF)
3. Oldest-Cell-First (OCF)

In WRR, the local buffer is granted access to the data node only once every t conflicts. This means each arbiter in the column is weighted using the address of the data node. This removes some unfairness from the equal round-robin algorithm. In LWF, each buffer and data node must also store the waiting time for each packet. The access to a node is decided by comparing the waiting times of two packets, and choosing the longest waiting time. In the algorithm used in this thesis for LWF, the wait counter is only incremented in the case of a conflict. In OCF (see also [69], time stamps between two packets are compared, and the smallest is always selected. When performing timing measurements with an HPD readout ASIC, these time stamps may be already available in the data packets due to the requirements of the application. If they are not present, they must be generated by other means. The drawback in OCF is that either the time stamp counter cannot be allowed to roll over during the acquisition or a roll-over must be detected. This problem is akin to the one found in real-time routers in communication networks [70]. Note that an algorithm such as longest queue first (LQF) (see [69]) may not be applicable here because the local buffer may have different size than the node buffer, thus rendering comparison of the queue occupancies meaningless.

Different data packets required by the data fabrics are shown in Figure 2.25. Naively, the WRR seems to have the least amount of overhead in terms of data in the packet. But this is only true if the architecture does not already include time stamping information as a part of its data. If it does, the data in the WRR and OCF are almost identical. The LWF requires an extra field for keeping track of the waiting time in each node and in the first slot of each buffer.

The node-based data fabric also has non-uniform latencies for packet transmission from different buffers. The latency T_n for a packet from a buffer n is thus

$$T_n = \frac{1}{R_{Bn}} + \sum_{i=0}^{n} \frac{1}{R_{Ci}} \tag{2.15}$$

If it is assumed that there is a data bus of constant width between each node, it can also be assumed that $R_{C0} = R_{C1} = \cdots = R_{Ck-1} = R_C$. Now if it is also assumed that there are no conflicts and the nodes and the buffers operate at the same speed, by simplifying (2.15), for a buffer n $T_{n\,min} = (n+1)R_C^{-1}$. This is the minimum achievable latency for packets, regardless of the arbitration scheme.

Figure 2.26 shows the readout efficiency E_{ro} for different arbitration schemes. It is assumed here that the number of buffers $k = 64$. No large difference is observed between the WRR and the OCF schemes, and the ratio of 1.5 is required to reach the E_{ro} of 99 % or higher. The LWF has slightly higher E_{ro} given the same ratio compared to two other schemes. The ratio to reach 99 % is 1.3. As LWF uses a local counter instead of a global notion of time required in OCF for the arbitration, the hardware implementation of LWF can be used even in applications without time stamping. The LWF also does not suffer from the limited dynamic range of time stamping.

Two bus-based column architectures using the token arbitration are also shown in Figure 2.26 for comparison. When the size of the inner token $G = 8$, a similar performance to the LWF scheme is observed. The difference is that, for a token arbitrated bus, the output rate R_C must be a global rate across the full column. Data must be sent from the top-most buffer to the bottom in time R_C^{-1}. This imposes tighter global timing constraints than when using the LWF scheme. In the LWF as in any distributed, node-based data fabric, the rate R_C is only a local requirement, and the timing constraints need to be met only between two nodes, not across the full column. For a column architecture with a token inner group size of $G = 1$, which is more similar in terms of the timing constraints to a node-based data fabric, R_C must be approximately two times higher compared to other architectures.

The minimum achievable latency for a given buffer address was presented earlier. The maximum latency for a given ratio of $\frac{R_C}{R_A}$ is obtained by simulation in this thesis. This ratio is set to 1 for these simulations. Note that if the ratio drops below 1, a shutter-signal can be used to control the amount of generated input data. Figure 2.27 presents the latencies of

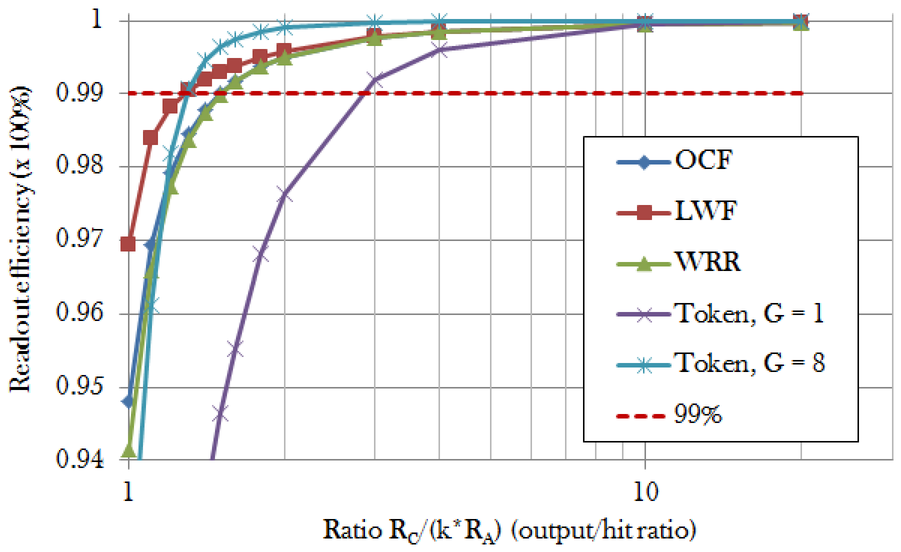


Figure 2.26: Readout efficiencies of the linear data fabric with different node arbitration schemes (k = 64).

different node-based data fabrics and a bus-based column with $G = 8$.

It can be seen that with $G = 8$ and the LWF, the latency profile of the two architectures is very similar. For the OCF and WRR schemes which perform worse in terms of E_{ro} than the others, a clear upper limit for the latency can be seen. This limit is < 128 for the OCF, and < 150 for the WRR. This is a desirable characteristic, especially if time stamping information is collected, because it indicates that the dynamic range of the time stamp does not have to be very large (8 bits) for $k = 64$ buffers per column. 8 bits of time stamp is enough to unambiguously identify the hits in time. For the OCF, an extra bit is needed to detect the rolling over of the time stamp counters.

As the last point, the scalability of the distributed data fabric is considered. Some reticle-sized HPD readout ASICs have already been designed and manufactured successfully [45, 37]. In chips of these dimensions, the column is typically more than 2 cm long. If a bus is utilized to send data from the top to the bottom, the data must be driven a long distance through a highly resistive and capacitive wire. Unless a tri-state bus is used, which has a higher wire capacitance requiring larger drivers than other bus structures, the gate delay further deteriorates the performance of the bus. Unlike in a bus-based architecture, adding more nodes to the distributed data fabric does not cause additional gate- or wire delays. However, it increases the latency in terms of number of clock cycles, and this increase must be taken

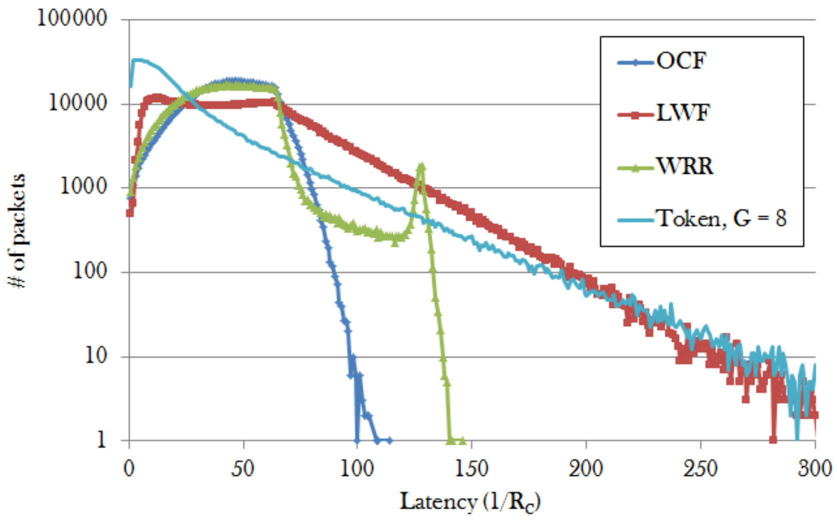


Figure 2.27: Latencies of the distributed data fabric with different node arbitration schemes. Latency is measured in periods of packet transfers (${R_C}^{-1}$).

into account when choosing the dynamic range of the time stamps.

The scalability can also be improved by deploying the super pixel technique presented in Section 2.12.3. Using this technique, several buffers or pixels are connected to a single node in the data fabric. This decreases the total number of nodes, and thus the minimum number of clock cycles required to pass through the fabric. Depending on the number of buffers sharing a node, the hardware overhead can also be reduced because the node register, the node arbitration and data multiplexing logic are shared among several buffers.

2.13.2 Hierarchical node-based data fabric

There are a number of techniques to reduce the latency of data coming from a node-based data fabric. To reduce the fixed, non-congestion -based latency, grouping of nodes can be used to create a hierarchical structure where the longest chain of nodes is no longer determined by the total number of nodes in the fabric. Using this grouping technique, packets sent from the last node in the fabric need not pass through all nodes anymore. In this part, only a fabric with two levels of hierarchy is considered, but in theory it is possible to divide a data fabric with k buffers and nodes into $log_2(k)$ levels of hierarchy. More than two levels is not practical for small pixel sizes, will be discussed in Chapter 4 where physical implementation is considered.

A node-based data fabric with two levels of hierarchy is shown in Fig-

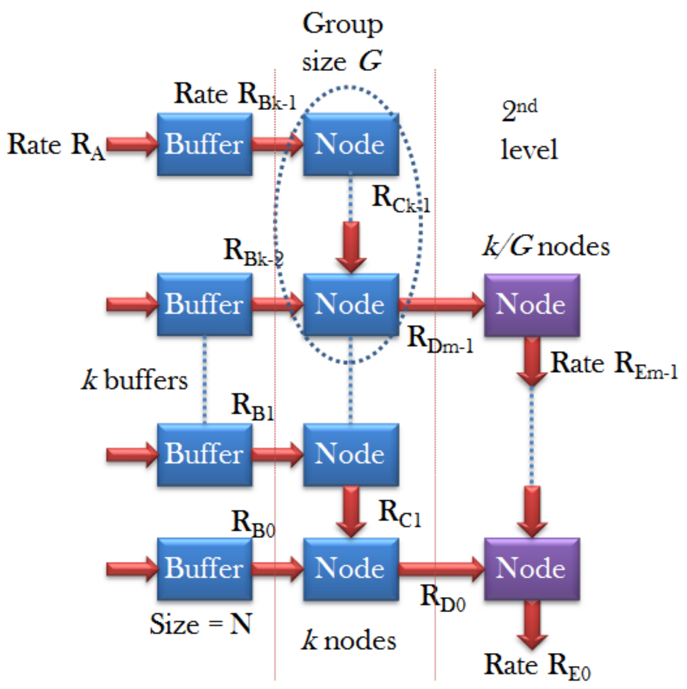


Figure 2.28: A node-based data fabric with two levels. The first level has k nodes and the second one $k/G = m$ nodes.

ure 2.28. Notice that each node has a similar structure to that shown in Figure 2.24. Again, to simplify the analysis, it is assumed that $R_{Ck-1} = ... = R_{C1} = R_C$, $R_{Dm-1} = ... = R_{D0} = R_D$, and also that $R_{Em-1} = ... = R_{E0} = R_E$. After this assumption, the minimum latency for data packets is $(G\ R_C + m\ R_E + R_D)^{-1}$. As in the case of the linear data fabric, the upper limit for the latency is set by the size N of the buffers and the arbitration algorithm within the nodes. If the arbitration algorithm grants access eventually to each node and buffer, this latency is upper-bounded. When N gets larger and the arbitration algorithm becomes less fair, the latency of packets increases in proportion. Note that the algorithm on each level of nodes can be chosen independently of each other, and the optimal choice of the arbitration algorithm may depend on the number of levels and the size of each group (G). Unlike in the case of the linear data fabric, only the WRR algorithm was used due to its simpler implementation compared to the other methods.

The performance of the two-level fabric was simulated using the same testbench as for the linear fabric. Again, address-wise uniformly distributed packets were injected into the fabric. The number of packets per an interval of time was generated using Poisson distribution. Figure 2.29 shows the

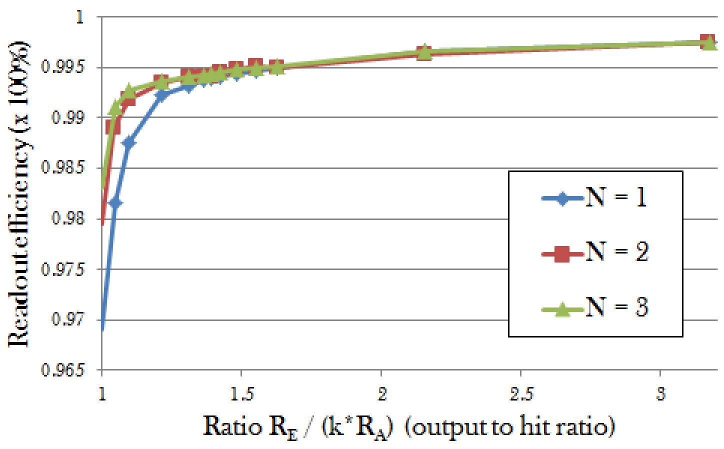


Figure 2.29: Readout efficiencies of the hierarchical 2-level data fabric ($k = 64$, $G = 4$) using WRR arbitration in nodes.

readout efficiency versus the ratio of input and output rates with three sizes of buffers ($N = 1, 2, 3$) and a group size $G = 4$. The ratio to achieve efficiency of 99 % is around 1.2 for $N = 1$, < 1.1 for $N = 2$ and < 1.05 for $N = 3$. The result for $N = 1$ indicates the hierarchical fabric can be even more efficient than a linear data fabric or a token-based bus with a group size $G = 8$ or smaller given the same output-input ratio. The difference in performance is approximately 10 %, but the physical implementation details may have an impact on the final choice.

The impact of the group size G to the efficiency was also estimated, and is shown in Figure 2.30. There is no large impact on the choice of the group size to the efficiency. Based on these results, it is always beneficial to optimize the group size for minimal latency in clock cycles. However, a larger group size G will have impact on the maximum clock frequency (and thus the latency) and on the throughput as will be seen in Chapter 4 where physical implementation details are discussed. On the other hand, if the clock frequency is set by the application due to timing resolution requirements, and a higher frequency clock is not available for readout, optimization of group size G can be done at no cost to maximum throughput.

The latency of the two-level data fabric was also estimated, and it is shown in Figure 2.31. It is clear, as discussed above, that the average latency is smaller in the hierarchical data fabric than in the linear fabric, when the group size G is optimal. When the buffer size $N = 1$, the worst case latency is reached in the event that the last packet of the last node loses the first arbitration in each node. As unlikely as this event is, the impact of bursts of arriving packets, which can cause this worst scenario, must be taken

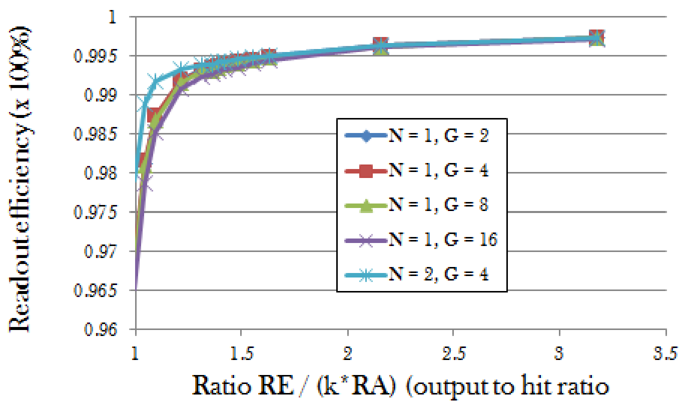


Figure 2.30: Readout efficiency with different group sizes G for 2-level data fabric ($k = 64$) using WRR arbitration.

into account if a specific minimum latency is essential for the application in question. In the case of $k = 64$, the smallest average latency is achieved using $G = 8$. The most probable values of the latencies for $G = 2$, 4, 8, 16 were 34, 21, 16 and 20, and were obtained from the same simulations as the efficiency figures. Based on these average latency values, it can be concluded that the hierarchical data fabric offers attractive options over the linear fabric for the latency reduction.

2.14 Network on-pixel chip

A novel way to transfer data from pixels to the periphery is to use a two-dimensional network, in which packets are sent between nodes. While widely used in network-on-chips (NoCs) for several years already, this technique has not been deployed in pixel chips in the past. This is an extension of the concept from the previous section, where data nodes communicated in a vertical direction only. In a fully networked system, the nodes communicate also in the horizontal direction. Vertical connections are implemented as unidirectional due to the structure and function of an HPD readout ASIC. In theory, it would be possible to distribute the configuration information to the matrix also using the same network, but in practice shift-registers are used for this purpose because high bandwidth is not required. All data must be first moved to the periphery of the chip before transmitting them off the chip as shown in Figure 2.1. On the other hand, allowing horizontal data flow in both directions gives more flexibility in avoiding highly congested nodes, and more importantly distributes the traffic more uniformly in the horizontal direction. The network is also more robust to manufacturing

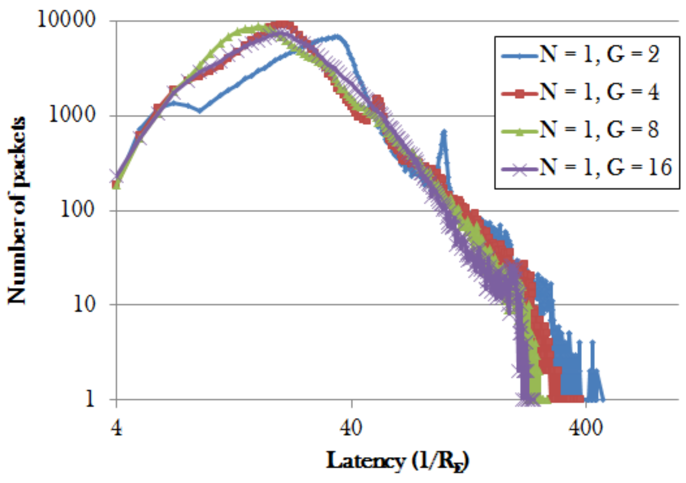


Figure 2.31: The latency of packets in the two-level data fabric.

errors and noisy pixels that cannot be masked because it allows routing of packets via neighbouring columns in the case of a broken data node in one column.

There are also some drawbacks in using the networked architecture compared to using a column-based approach only. The network requires routing of digital signals over analog areas. This may cause coupling of signals from digital area to the analog. As [71] mentions, a 2-V swing in a digital line coupled to the analog via a capacitance of only 0.5 fF can inject a charge of 1 fC to this node. The networked system requires a larger area for arbiters and controllers than a column-based system because the arbitration must be done for all inputs and outputs. Each data packet in the network must also contain the full address of the source of the packet. This increases the number of bits per memory slot by log (x) The network of $k \times x$ nodes is shown in Figure 2.32. The analysis of this network needs to take into account more things than in the case of the data fabric in the previous Section 2.13 for several reasons:

1. Data can move in three directions instead of one.
2. Each rate $R_{i,j}$ has an impact on a two-dimensional set of nodes. More precisely, the rate $R_{i,j}$ can impact nodes $N_{i',j'}$ where $i' = 0...x - 1, j' >= j$.
3. There are 4 inputs and 3 outputs for each node instead of 2 inputs and 1 output.
4. Analytically choosing arbitration priorities cannot be done without

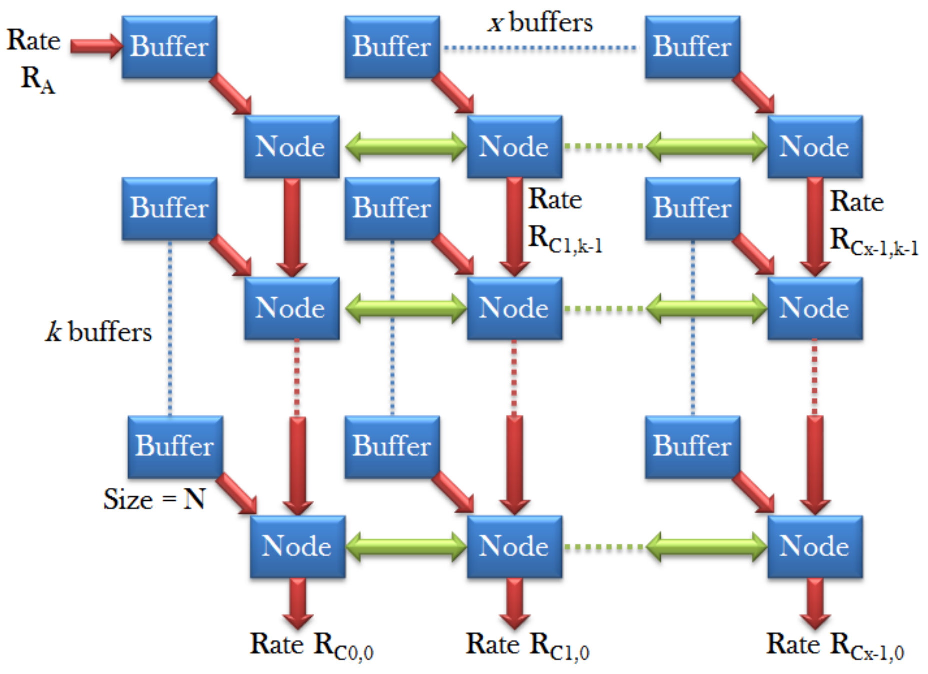


Figure 2.32: A network of $k \times x$ data nodes.

full analysis of the two dimensional network. In this thesis, optimal priorities are selected based on empirical simulations results.

The network must have at least an output rate

$$R_O = \sum_{i=0}^{x-1} R_{i,0} >= kx \times R_A \tag{2.16}$$

For simplicity it is assumed that $R_{C0,0} = R_{C1,0} = ... = R_{Cx-1,0} = R_C$. This assumption holds without loss of generality if the periphery has, for example, a bus with fair arbitration. After the assumption, (2.16) can be restated as $R_O = R_C >= k \times R_A$.

A more detailed view of a single node of the network is shown in Figure 2.33. The node consists of a storage register for one packet, an input arbiter and an output arbiter. The input arbiter selects the next input data source for the register if the register is empty. Similarly, the output arbiter chooses the next destination for a data packet in the register if any are available. As each arbiter must perform a handshake with adjacent nodes, there is an overhead of several clock cycles for arbitration which cannot be utilized for sending data. As is customary for network directions, four cardinal directions are used to denote different directions in the network.

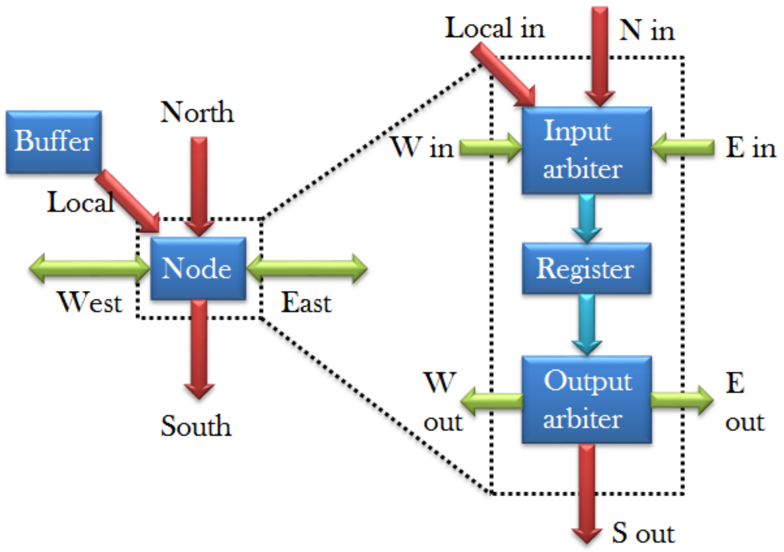


Figure 2.33: A close-up of one data node of the network and a local buffer.

Unlike in the previous sections, the following simulations were performed using synthesizable RTL descriptions instead of non-synthesizable models. The main reason for this was that to simulate accurately the behaviour of data nodes a clock cycle accurate model was needed. This is important because the data transactions between the nodes can take several clock cycles, and the transactions can be interleaved in time. This means that when a node has sent the first word of a packet to the next node, it can already read in one word from another node. Thus, a data packet can have its words split between two data nodes.

Unlike the previous architectures, the network has essentially two dimensions, and thus the results are presented as two dimensional matrices instead of scatter plots. Figure 2.34 shows the readout efficiency per data node and Figure 2.35 the average latency per data node of all measured packets in a network of 64×64 nodes. The hits are randomly and uniformly distributed with a total input rate of $64^2 \times R_A = R_O$. The WRR arbitration introduced in the previous section is assumed for local versus non-local data. Non-local data arbitration works as follows: Select north input if it is available. Otherwise, when in conflict between east and west, select the next input in a round-robin fashion. This arbitration is done in the input arbiter. The output arbiter always forwards the packet to south if possible. Otherwise, it selects from east and west (if both are available) in a round-robin fashion. To avoid useless live locks and bouncing of packets between two data nodes, a packet cannot be sent back in the direction from which it came unless it has been routed to south at least once between transfers. This requires keeping the direction of the last arrival in a memory in the data node (one extra bit) but does not require any extra wires for connection between nodes.

The efficiency is higher on top of the columns because there is less congestion in the data nodes on the top. The local buffers in the data nodes are able to access the data network faster due to the WRR-arbitration. As all packets must be routed through the nodes at the bottom of the column, these are areas with more congestion. The local buffers at the bottom of the column thus have lower frequency of access to the network. The latency is shown as the number of periods of ${R_C}^{-1}$. The latency is as expected, higher toward the top of the columns. On the top of the matrix, the latency is directly proportional to the row address of the data node.

This proportionality is lost lower down the matrix. For example, packets from the first (lowest) row can still have a latency of up to $13 \times R_C$ which is higher than the row address. This is a result of the WRR arbitration algorithm (see Section 2.13) which assigns more bandwidth to the data coming from the top.

Note that these plots are not used to make direct comparisons to the column-based data fabric. They illustrate the expected characteristics of the network only, and the trends in the efficiency and latency.

To assess the performance of the network compared to a column-based, vertical-direction-only data fabric, the horizontal routing was disabled and the results compared to a system where this routing is enabled. The comparison of two systems using randomly and uniformly distributed hits is shown in Figure 2.36. The distribution of hits is shown in the plots on the left. The readout efficiency is shown in the plots on right side. The only noticeable difference can be seen in the region at the bottom of the pixel matrix. When horizontal routing is disabled, the first few rows have much lower efficiency than with the horizontal routing enabled. Because enabling the horizontal routing allows packets to avoid hot spots and congested areas more easily, packets arriving from the top do not need to utilize their column of origin only. The simulated total efficiency without east-west routing is 91.4%. When the routing was enabled, the overall efficiency across the full matrix is 92.6%. The used output-input rate ratio in the simulations was equal to 1.

When using a non-uniform distribution (Gaussian with a mean of 16, standard deviation 4), the difference of the efficiencies of the two systems is more significant. This is illustrated by Figure 2.37. The overall efficiency without east/west-routing is 76.9% and correspondingly with the routing enabled 81.5%.

Finally, Figure 2.38 shows the total efficiency of three different 64×64 network simulations. “No pixels” -label indicates that each node consists of

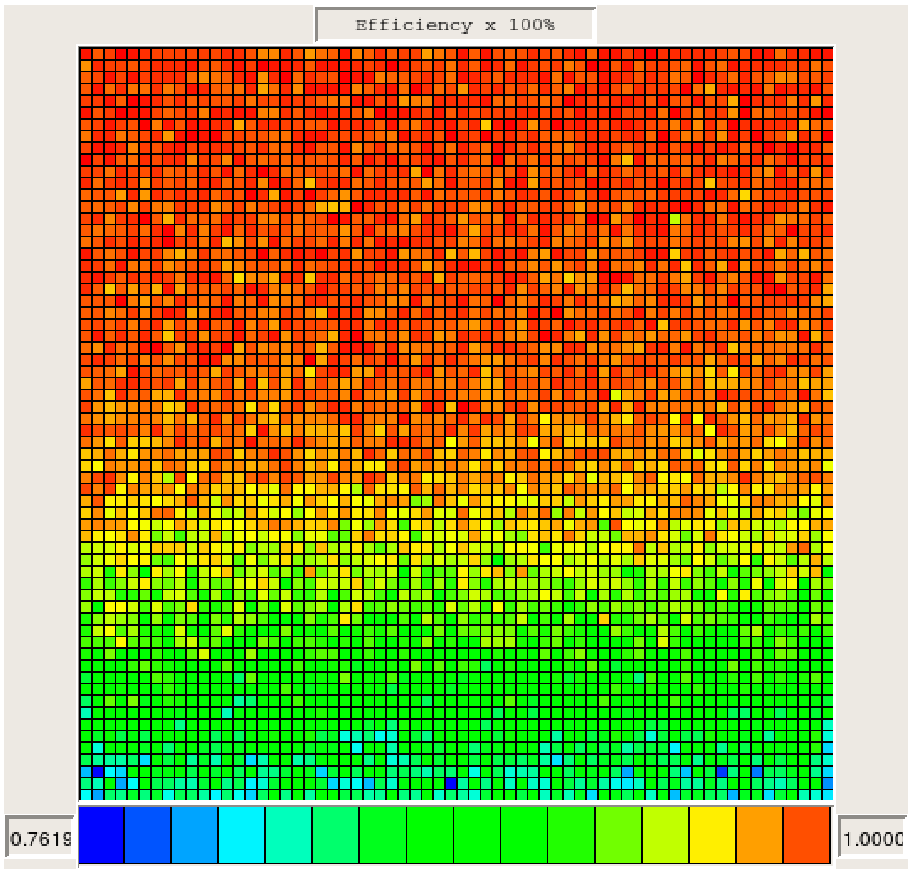


Figure 2.34: Readout efficiency per data node of a 64 × 64 network.

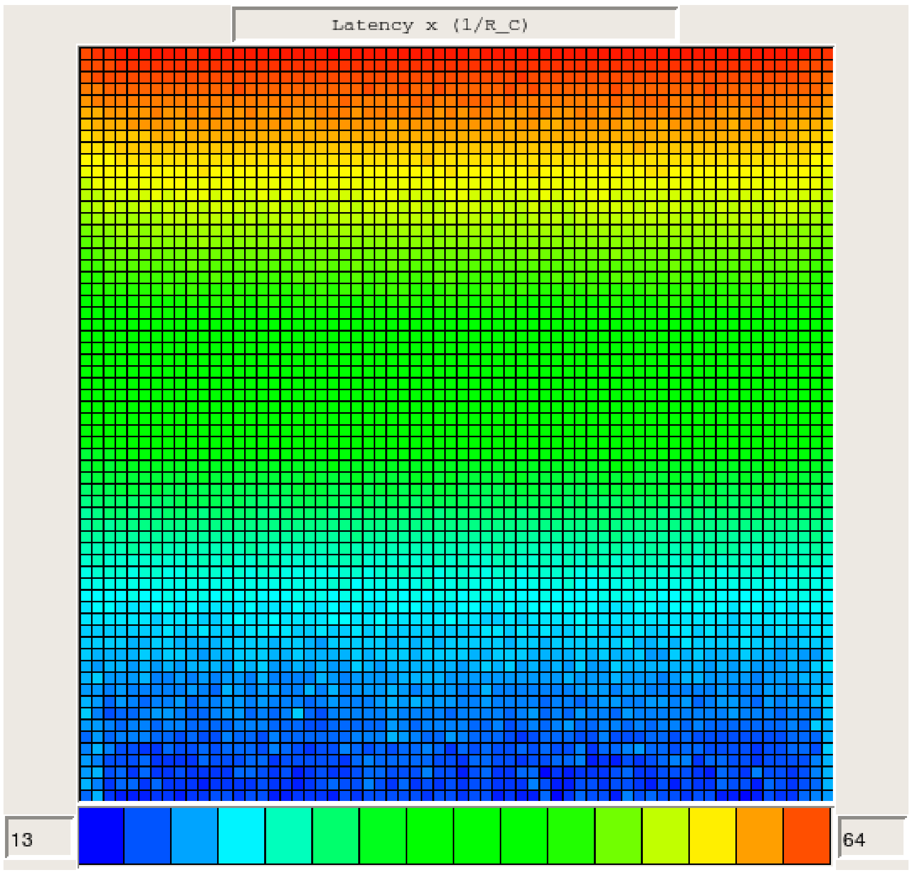


Figure 2.35: Average latency (in ${R_C}^{-1}$) per data node of a 64×64 network.

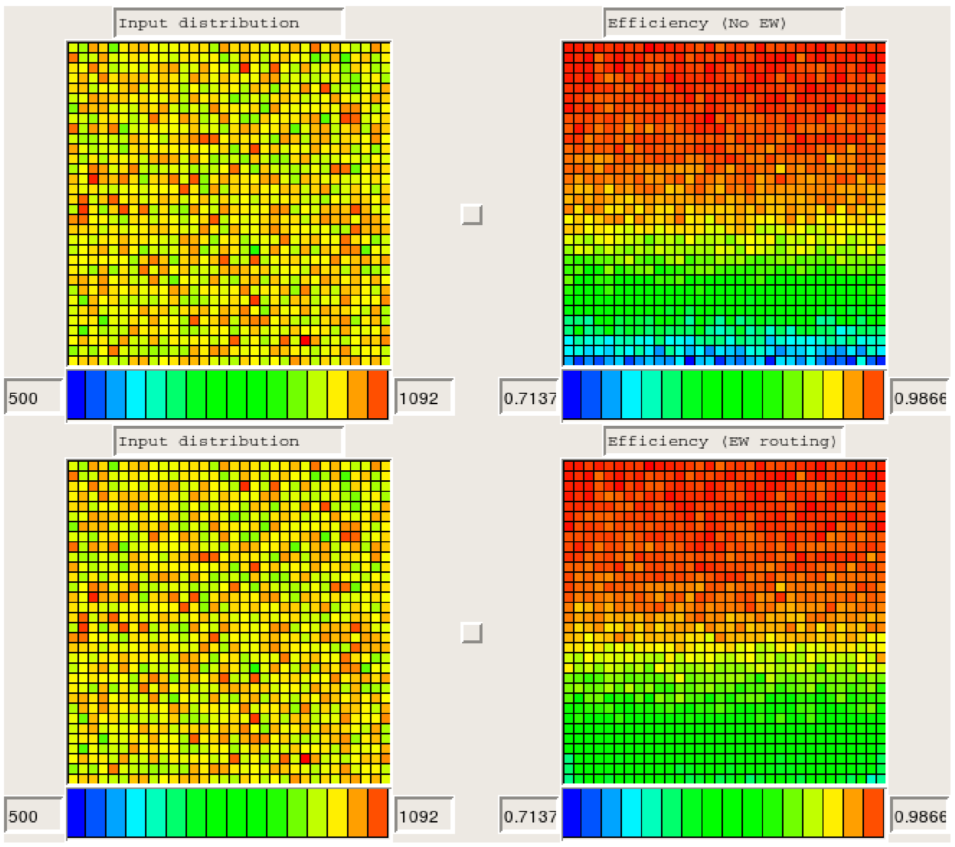


Figure 2.36: Comparison of two 32 × 32 networks. The top row does not have east/west routing, and the bottom row does.

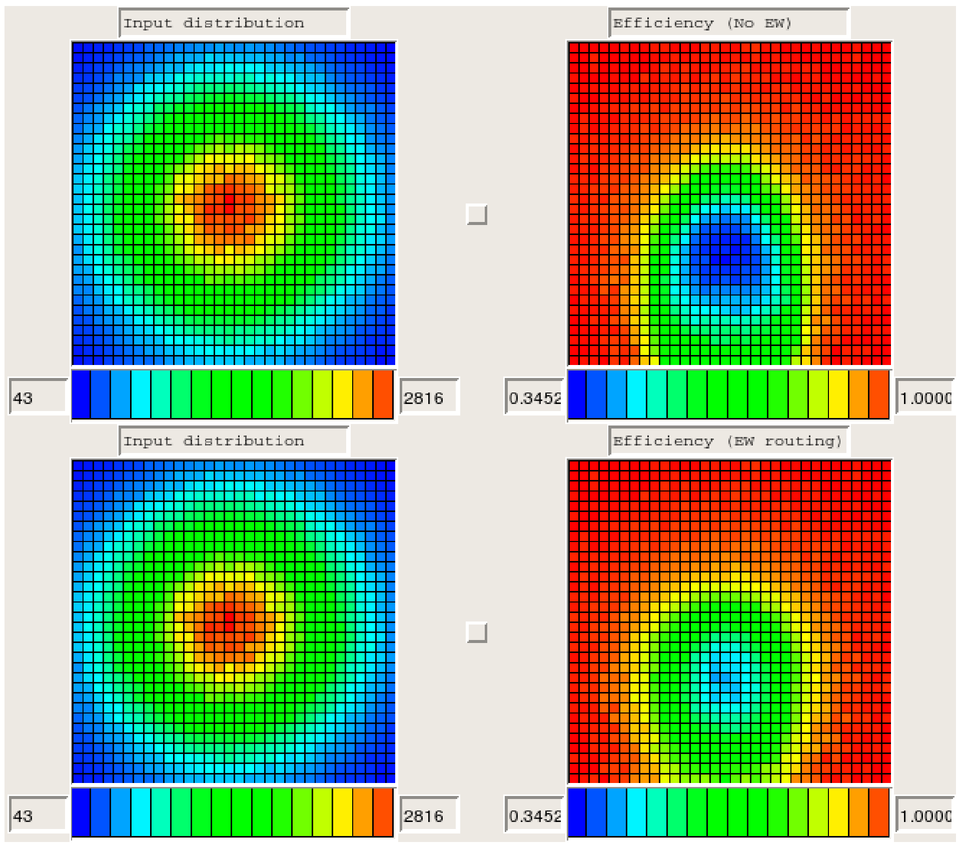


Figure 2.37: Comparison of two 32×32 networks. The top row does not have east/west routing, and the bottom row does. The hits are non-uniformly distributed.

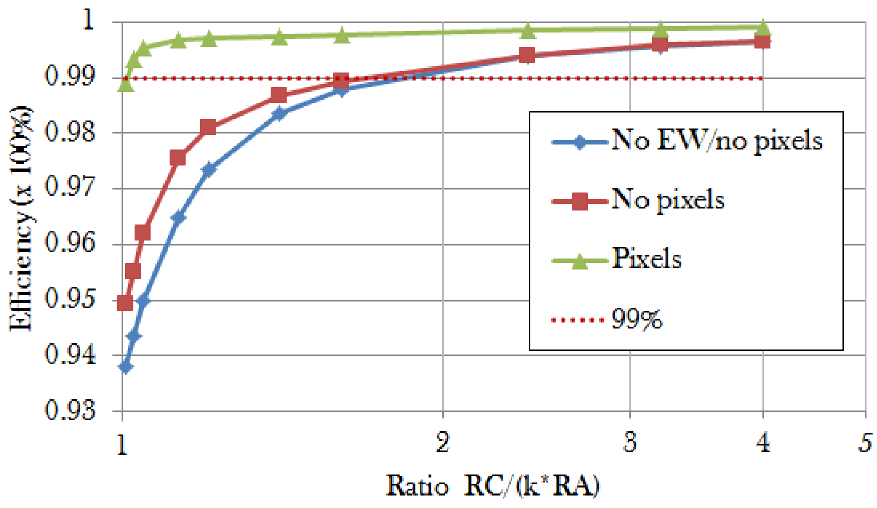


Figure 2.38: Comparison of efficiency of three 64 × 64 networks.

only one buffer without pixels being modelled. When the east/west routing has been disabled, the efficiency is lowest at low ratios. When the ratio $\frac{R_C}{k \times R_A}$ exceeds 2, the benefits of east/west routing start to be negligible in terms of the efficiency. As expected, given the same input rate, this reduces the efficiency compared to a case where pixels are modelled. In this case 8 pixels per data node are assumed.

Comparing the efficiency of the 64 × 64 network with pixels to previous readout architectures, the network architecture has the highest efficiency with the lowest output/input ratio. At approximately a ratio of 1.02, the 99 % efficiency mark is hit. This means there is very little extra bandwidth required to achieve a certain rate, thus reducing the power consumption and requiring less hardware resources such as wide parallel buses.

In this section, it has been demonstrated that it may be advantageous to deploy a network of nodes using two-dimensional connections instead of moving data to one direction only. In the later chapters, an analysis of the additional hardware resources required in comparison to a column-based, vertical-direction-only solution is performed. Due to area constraints imposed by pixel dimensions, the feasibility of implementing this kind of network can be evaluated only after the full PnR of the design. The crosstalk and coupling of digital signals routed across analog pixel area can only be fully characterized from a manufactured device.

The network techniques and the data fabric techniques can also be combined. For example, [72] presents a hybrid structure of global network with local communication using rings and buses. This can be useful because

Table 2.1: HPD ASICs with time/charge measurement. FF = full frame, SEQ = sequential readout, PKT packet-based, TRG = triggered, CO = continuous, ZSF = zero-suppressed frame, ToT = charge measurement, AN = analog pulse height

Name	Tech.	Cols $\times$ Rows	Pixel μm^2	Readout	Time resol.	Ref.
Alice1LHCb	250nm	32x256	50x425	FF,TRIG	25ns	[47]
CMS 0.25μm	250nm	52x80	150x100	PKT,TRG,AN	25ns	[48]
FEI-2	250nm	18x160	50x400	TRG,	25ns	[74]
FEI-3	250nm	18x160	50x400	PKT,TRG	25ns	[49]
Timepix	250nm	256x256	55x55	FF,SEQ,ToT	10ns	[25]
FPIX2	250nm	22x128	50x400	PKT,CO	132ns	[54]
FEI-4	130nm	80x336	50x250	PKT,TRG,ToT	25ns	[75]
TDCPix	130nm	40x40	300x300	PKT,CO,ToT	150ps	[45]
CLICpix	65nm	64x64	25x25	ZSF,SEQ,ToT	10ns	[35]

the local hit rate gradients in a pixel chip are rarely non-uniform but the hit rate gradient across the full pixel matrix can change. Because the two-dimensional network is best suited for equalizing the traffic globally, and the unidirectional data fabrics require less area and are simpler to implement, a mixture of these techniques could be adopted to arrive to the most optimal solution in terms of throughput, latency, power and area. In general, as [73] states, the optimization of this network only makes sense for a specific traffic distributions, and source and destination address distributions. Unlike in generic networks, in the case of a pixel chip only the source address (input hit) distribution has to be considered because there is no necessarily fixed destination address for the data. This last point also ensures that the routing algorithm need not know the destination addresses.

2.15 Previous HPD ASICs

In Table 2.1, previous HPD ASICs related to this work are listed. Only chips capable of timing measurement and in some cases also charge measurement are listed. The pixel size is given as height (Y) $\times$ width (X).

In addition to the chips described in Table 2.1, several other HPD ASICs exist [42, 76, 77, 43, 44]. These are mainly frame-based readout chips developed for very high occupancy (close to 100%) and for hit rates $> 1GHz/cm^2$, and for imaging applications with event counting and single photon counting capabilities. The chips have no timing measurement capabilities, and are summarised in Table 2.2.

Table 2.2: HPD ASICs without time-of-arrival measurements. FF = full-frame, CO = continuous

Name	Tech.	Cols $\times$ Rows	Pixel μm^2	Readout	Ref.
Dosepix	130nm	16x16	300x300	FF,	[77]
Eiger	250nm	256x256	75x75	FF, CO	[76]
Medipix3RX	130nm	256x256	55x55	FF, CO	[42]
Mönch	130nm	160x160	25x25	FF	[44]
XPAD3	130nm	80x120	130x130	FF	[78]

2.16 Prototype chips

A number of HPD readout ASIC prototypes have also been designed. This section lists some of these relevant to this thesis. Because the architectural challenges for small prototype chips are not the same as for full-fledged systems, a full analysis of their capabilities is not done. They may still have interesting architectural properties that can be scaled to full-sized chips.

In [46], double column buses are implemented as differential lines using reduced swing pseudo differential CMOS signaling. The advantage of this approach is lower power consumption than when using the full CMOS swing. Due to the reduced swing, the speed of the bus is also improved. The drawback of this method is that the implementation is full-custom thus excluding the possibility of automatic static timing analysis (STA) tools. It is mentioned in [46] that due to a connection error and too high a capacitive load, the bus could not be run at targeted frequency of 160 MHz, and the speed was reduced to 50 MHz.

2.17 Concluding remarks

This chapter introduced several concepts related to HPD readout ASICs. An overview of the floorplan of an HPD readout ASIC was given, in which the chip is divided into two main parts: The active pixel area or the pixel matrix, and the periphery. A definition for readout efficiency of an architecture was given, and the occupancy was presented as a measure for the activity within the architecture.

A novel time stamping method was presented to reduce the switching activity in global time stamp busses spanning the full pixel column. This method also reduced the number of wires required for the distribution of the time stamp to two. As a trade-off, the contribution of the time stamping to overall digital dead time was increased in proportion to the time stamp size.

Different architectural solutions were compared using C++, SystemC and SystemVerilog RTL simulations, and two novel architectural solutions presented. Especially, two different data fabrics with locally communicating nodes were given. It was shown that these architectures could be used instead of the often used bus-based architectures without worsening the performance in terms of efficiency and latency. As the second novel architectural option, a full network inside the pixel matrix was proposed. Although the network has some implementation issues such as routing of digital signals over analog sections, it offers an option to improve the yield of the system by routing data packets around digital sections with manufacturing defects instead of having a full unusable column. It was also shown by simulations that in the case of non-uniform occupancies, the network equalizes the data traffic among all columns in the pixel matrix. The benefit of this is that the architecture need not be designed to run at higher frequency or rate to accommodate for local hotspots.

A list of existing HPD readout chips was compiled at the end of the chapter, and the chips were characterized based on the definitions presented in this chapter. From this characterization it could be seen that no pixel readout chip with nanosecond-level measurement capabilities with pixel pitch in the range of 55 μm exists. This is one of the gaps that one of the architectures presented in this thesis tries to address.

Chapter 3

Simulation of HPD ASICs

This chapter describes the modelling and simulation techniques used during the work done for this thesis. It also presents technical details of how the simulations and the models were implemented by the author. The purpose of the modelling in this thesis was to guide the design toward a specific architecture meeting all requirements, which could then be implemented using logic synthesis and automated PnR. It also enabled the exploration of several architectural options without having to go through more time consuming RTL design, debugging and functional verification stages. Different architectural concepts were explored in Chapter 2, and the results were used as the starting point of the simulations. A final goal of the simulation implementation was to allow the use of realistic test vectors generated by software descriptions of the different detector applications.

As described in Chapter 2, there can be tens of thousands of pixel channels per HPD ASIC. Modeling each pixel very accurately at transistor-level or even at RTL can increase simulation run-time drastically when compared to higher level models. Analog front-ends can be modelled behaviorally as well as the digital, with some details being simplified. For the rest of the chapter, a word 'model' refers to a non-implementation level, non-synthesizable model unless explicitly stated otherwise.

3.1 Simulation tools and methodologies

3.1.1 Methods in this thesis

This thesis uses RTL, TLM and behavioral abstractions for modelling the hardware in simulations. RTL models are synthesizable and use bit-accurate communication. TLM components communicate using transactions such as pixel packet transactions and request-transactions for arbitration. Behavioral models use a predetermined data structure for inter-component communication such as a data packet modelled as a class.

Optimal sizes for super pixels for achieving the highest data reduction are determined using functional, non-timed simulations. A simple address mapping algorithm for this is described in Section 3.2.3.

C++/SystemC and SystemVerilog (SV) are used as modelling languages. SystemC has been used because of the flexibility and efficiency of C++ and the access to free open-source simulator. The main reason for the deployment of SV has been the usage of Open Verification Methodology (OVM) and Universal Verification Methodology (UVM) libraries which also contain implementations of TLM 1.0 interfaces (UVM has also TLM 2.0). These libraries contain useful functions like factory-method implementation which allows a fast iteration of different models without changes in the existing models by changing one line of code in the top-level test class which instantiates the simulation models.

Also, no analytical models are used in the simulations. The main motivation for this is the complexity of deriving new models, and the difficulty of introducing new parameters to the analytical model without fully re-verifying and proving its validity. Using analytical models for front-end buffers and other simple components also makes it more difficult to use them as base classes for more complex models.

3.1.2 Methods used in related work

Some simulation and modelling work has been carried out related to HPD readout ASICs. This section briefly summarizes previously published works.

In [40], a statistical approach is used to determine optimal sizes for a super pixel to group multiple pixel hits into one packet. When a set of pixel addresses is given to the algorithm, it computes how many packets are created based on the size of a super pixel. The algorithm has a second layer which performs a bit-level encoding for the hits and supports multiple encoding schemes. In this way, the most efficient super pixel area and bit-level encoding are found.

In [79], a generic pixel simulation framework VEPIX53 is presented. It has been implemented using SV and UVM. Using this framework, behavioral simulations of two super pixel buffering architectures are presented at column-level level. No full chip simulations are presented.

One approach to simulating a full pixel chip has been to develop a non-synthesizable very high-speed integrated circuit hardware description language (VHDL) model and integrate this with a Monte Carlo (MC) hit generator [36]. Because SV offers all required constructs for high-level simulation, no VHDL was used during this thesis.

In [37], a high-level C++-simulation framework has been developed. It was used to study inefficiencies related to the pile-up at the front-end and the overflows at the super pixel buffer. An integration with physics event data has also been done in [37]. It is also reported in [75] for the same project, that some analytical models for the inefficiency calculations were used.

An RTL simulation approach has been taken in [38]. They report a simulation performance of 2800 clock cycles in one second of real-time. It is described that the simulation contains 16 readout chips (52×80 pixels) with a total data rate of 320 Mbps. However, it is unclear from the reference whether a full chip RTL model has been used in this case.

A study of a 256×256 pixel matrix architecture [39] has been carried out using a combination of C++ modelling and VHDL simulation. No comparison of execution times between the C++ and VHDL models is given.

3.2 Architectural Simulation

In this section, several techniques used for architectural simulations are explained. The main goal of these simulations is to find suitable architectures for HPD readout ASIC architectures targeting specific applications. Another aim is to create a suitable framework for comparing and studying different architectural solutions.

Due to the large amount of parallel channels and increasing complexity inside a pixel, simulation of a full HPD ASIC can be accelerated substantially by modeling the chip above transistor-level or even above RTL. The run-time improvements gained from high-level modelling approaches are presented later in this chapter.

Gajski and Cai [80] present nine different levels of abstraction for a system. They divide the functionality into communication and computation. These components then have three different levels of abstraction: untimed, approximate-timed and cycle-accurate.

Based on the definitions in [80], three different levels of abstraction are used in this thesis:

- Sequential, cycle-based communication, untimed computation (C++, SV)
- Approximate-timed communication and computation (C++, SystemC, SV)
- RTL simulation with synthesizable circuit models. (SV)

In a sequential model, the shortest simulation interval equals one cycle. Communication latency is measured in these cycles and events shorter than one cycle are not simulated. In this thesis one cycle is matched to the clock cycle of a system but this need not be always the case. There is no notion of time except a counter for calculating the number of executed cycles. This kind of simulation is similar to instruction-set simulators (ISSs) which can be, and are often, written in sequential programming languages such as C or C++. This kind of simulation can be executed in a single loop for a specific number of cycles. The sequential model can also be implemented without coroutines [81] because there is no need for event scheduling and support for multiple processes. A coroutine is a function which can have multiple entry points instead of executing in a strict sequential manner without exiting and re-entering the function until it is finished. The sequential model, however, is not always faster to execute than an approximate-timed model. This depends on the number of processes executing in parallel in a timed model. Also, if a sequential simulation performs a lot of useless computations in the execution loop, it can run noticeably slower than a timed model. A useless computation is a non-state altering functionality. For example, if each pixel is polled for data during each execution cycle while a small fraction of pixels hold data at any given cycle, this can slow the simulation down. Thus, a more efficient solution is to keep a list of hit pixels, and only poll this list during each cycle.

Approximate-timed models contain delays implemented using multiple processes executing at the same time and wait-statements, and so a simulation kernel supporting coroutines and process synchronization is required. As the simulation kernel is usually concurrent rather than parallel, there is only one active process at once. This means that each process must yield control to the kernel, otherwise that process will keep executing without giving control to other processes. This is called cooperative execution. To decrease the simulation run time of a timed model, computation and processing should only be done when there are data available. This means that the evaluation of states of the components is not performed at every clock cycle. In fact, the clock can be completely removed from approximately-timed models and synchronization of processes performed using mutexes, mailboxes, semaphores and events other than frequently occurring clock edges. SystemC [50] and SV [82] both have support for all these mechanisms.

RTL simulations can generally be executed using the same simulator kernel as approximate-timed models. The difference is that they are described by following a set of rules that make the descriptions synthesizable. It is also possible to create an optimized simulator for RTL which does not support timed simulations [83]. This optimization improves the run-time of simulations considerably, and it has been reported that it can execute at least at double the speed of a simulator supporting timed models fully. For a pixel chip with tens of thousands of pixels containing digital processing logic, RTL simulations can take hours to collect sufficient amount of statistics. This often means tens or hundreds of hits per one pixel in one simulation. Based on this very rough estimate, it can be quickly calculated that for a 65k pixel chip, a magnitude of 10M hits are required in total.

A generic algorithm for sequential cycle-based and approximate-timed simulation of the readout architecture of HPD ASICs has been developed. This algorithm is specific for an architecture operating without a trigger-signal. It consists of the following steps:

1. Generate hits for the current cycle or read them from an external file.
2. Inject hits into pixels.
3. Check the pile-up at front-ends.
4. Apply a grouping algorithm (clustering) for single hits to form 'clusters'.
5. Check the availability of buffers at the front-ends (if any).
6. Do arbitration of all column buses (if any).
7. Read a hit from each column if it is a readout cycle.
8. Check the state of EoC buffers for each column. If a buffer is full, write hit back into its column.
9. Arbitrate next EoC buffer(s) with data available to read.
10. For each output link, read corresponding number of bits (or hits) from the EoC buffers.
11. Repeat from the step 1 until enough cycles have been simulated.

There are some steps which need to be executed in a strict sequential order. For example, the grouping algorithm must be applied after checking the pile-up. Other steps, such as output link simulation can be done at any point. The algorithm described here can be further refined to include functionality like trigger-signals. Also, the dead-time of front-end buffers can be modeled similar to the front-end pile-up. In addition, because many of the steps are orthogonal, they can be simulated or even modeled separately. For example, the pile-up at the front-ends is proportional only to the hit injection rate and can be simulated at a very early stage without the rest of the model being implemented. The algorithm can be implemented in a fully sequential manner or using coroutines and concurrent simulation techniques.

3.2.1 Hit extraction and generation

The input stimuli for readout architecture simulations can be generated either by an external program or by the hardware simulation itself. A method used in this thesis is to first analyze a data set generated by a program

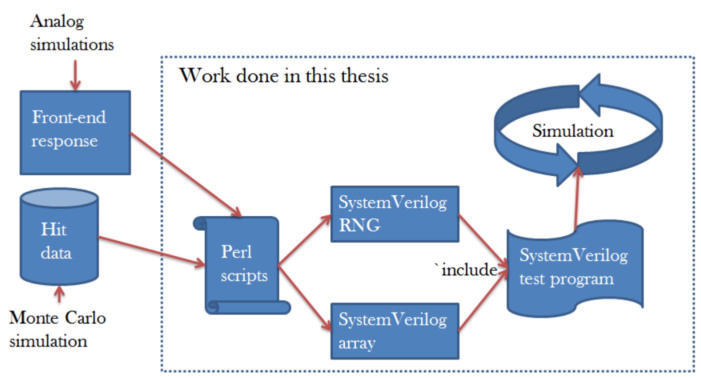
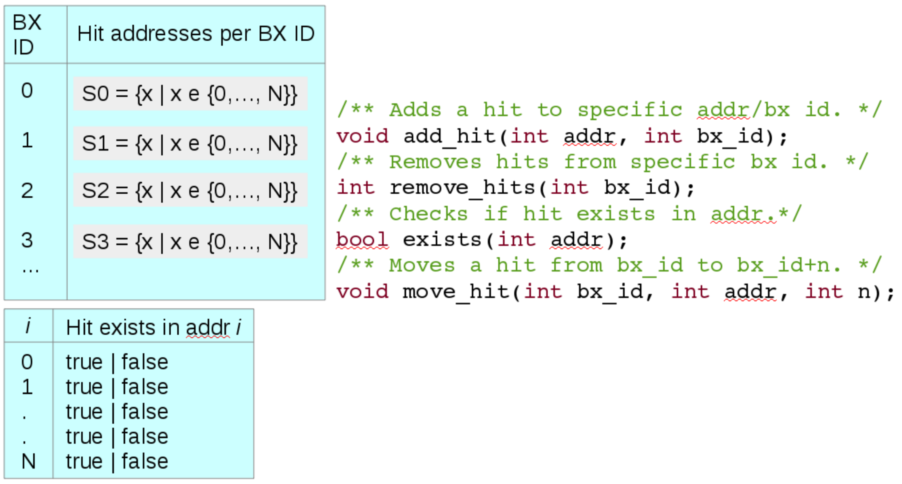


Figure 3.1: The process for extracting hits for the simulations.

external to the simulator, and create a weighted random number generator (RNG) based on these values. This approach requires less work if, for example, Monte Carlo physics simulations already exist for a specific application than trying to accurately model the physical phenomena creating the pixel hits. Because SV supports constrained randomization and generation of arbitrarily weighted distributions, it has been used to create weighted RNGs in the application specific simulations done in this thesis. Then, instead of using the original Monte Carlo data, these RNGs are used to verify the architectural performance. The advantage compared to the original data is, that by changing the seeds of these RNGs, different randomized scenarios can be created instead of repeating the same patterns of hits in every simulation run. Using application-specific Monte Carlo data as a basis for RNGs gives a more accurate simulation of the expected hit occupancy and architectural data traffic than randomization without using Monte Carlo data.

The process of hit generation is shown in Figure 3.1. Data are first produced using Monte Carlo simulations. These simulations do not contain a single monolithic model but consist of a set of layered generators from different physical phenomena, for example the charge generation mechanism in a silicon pixel detector. The important thing for architectural simulations is that the output of Monte Carlo simulations contains either a pixel coordinate or spatial information which can be converted to pixel addresses that match a specific pixel geometry. The data should also contain charge deposited per pixel for a pile-up simulation of the analog front-ends.

For the architectures designed in this thesis that are targeted towards a particular application, a Perl script was written to extract the address and digitized charge information from the Monte Carlo data. A simple charge-to-ToT converter was written as a Perl module to perform the TDC before actual hardware simulations. The characteristics of this converter



BX ID	Hit addresses per BX ID
0	S0 = {x | x e {0,…, N}}
1	S1 = {x | x e {0,…, N}}
2	S2 = {x | x e {0,…, N}}
3	S3 = {x | x e {0,…, N}}
…	

```
/** Adds a hit to specific addr/bx id. */
void add_hit(int addr, int bx_id);
/** Removes hits from specific bx id. */
int remove_hits(int bx_id);
/** Checks if hit exists in addr.*/
bool exists(int addr);
/** Moves a hit from bx_id to bx_id+n. */
void move_hit(int bx_id, int addr, int n);
```

i	Hit exists in addr *i*
0	true | false
1	true | false
.	true | false
.	true | false
N	true | false

Figure 3.2: A class for modeling front-end pile-up.

were obtained from transistor-level analog simulations. The Perl script was used to produce an SV RNG and two arrays containing address and ToT information. These files were then included and compiled with models before the simulation. The advantage of an RNG is that it can be reseeded instead of using the same list of hits over and over again.

For simulations of architectures with no specific targeted application, hits were generated mainly using uniform random distributions for spatial information and exponential distributions for times of arrival. SV has built-in RNGs for these distributions, and they were created for SystemC simulations by hand.

3.2.2 Front-end pile-up

The front-end pile-up was simulated to determine the response time or return-to-baseline required for the analog front-end and the conversion time for the digital front-end. This can be done using a behavioral, untimed model of the front-end. Figure 3.2 shows the model used for this simulation. The model contains functionality to add a single hit, remove all hits in one time ID slot (referred to as bunch-crossing ID or bx_id), and to check if a hit with a given address exists.

A lookup table is used to store all pixel hit addresses related to a specific simulation cycle. Each simulation cycle with hits in it contains an entry in a lookup table which consists of a set of pixel addresses (Hits per BX ID). A key to the lookup table is simply the time ID number. An important point to note here is that the time ID number indicates when hits are removed from the lookup table, and not when they were placed into the table. At each time ID interval, remove_hits() must be queried and the corresponding entry will be removed. To improve simulation speed, the model also contains a fast-lookup table to check if a hit for specific address already exists. Without this table all sets would have to be searched for an existing hit, making the algorithm considerably slower and run-time dependant on the size of these sets. With the fast lookup-table, an existing hit can be found in a constant time.

The last method shown in Figure 3.2 is implemented to move a hit in a given address from a specific time ID slot to a slot which is n slots forward in time. This models the pile-up of a secondary charge in the analog front-end, and indicates that the return-to-baseline of the front-end is extended because of this charge injection.

3.2.3 Hit grouping and clustering

As studies have shown [75, 40], grouping several pixel hits into one data packet can reduce the total data rate produced by the chip. Generally this reduction depends on the angle of the particle tracks, and hence the average size of the pixel clusters, and the amount of information that needs to be recorded per pixel. If each packet coming off the chip must have the same length, this can introduce overhead if a lot of unused information needs to be included in the packet. Consider the following example:

For each single pixel hit, assume that each hit consists of the following information: 16 bits for the address, 14 bits for a time stamp and 14 bits of additional payload. For single, non-grouped hits the total amount of bits is $44 \times N$, where N is the number of hits. If it is assumed then, that this information is grouped into a packet of 8 pixels, the address (reduced to 13 bits from 16 bits) and the time stamp can be shared. For the grouped packet, the total number of bits is $13{+}14{+}8{\times}14 = 139$. It can be calculated that when $N > 3$, the grouping is a more effective strategy. If the same exercise is repeated for 4 bits of information per pixel, the grouping is more effective than using single pixel hits when $N > 2$.

In this thesis, an LUT is used to perform the hit grouping. A function is created which maps each single pixel address into a region (group) address. The LUT is initialized by calling this function once for each pixel address and caching the values for later use. The region addresses are obtained from the pixel address using the following formulas:

Region row address = pixel row address / region Y size

Region column address = pixel column address / region X size

During each simulation cycle, this mapping is performed for all injected hits. After the regions have been determined for each hit, another function is called to encode all pixel hits within the region into a number of bits. In the case of a fixed length packet, encoding is trivial multiplication operation.

If packets can have different lengths depending on the number of hits in them, a function computing this number must be created. This kind of two-pass method allows an independent evaluation of grouping and encoding techniques.

3.2.4 Front-end buffering

Due to the limited area available in the pixel matrix, digital front-ends and super pixels do not typically have more than two levels of data buffering [48, 49, 75]. The first level is implemented at the pixel level and the second buffer at the super pixel level. It was already shown in the previous chapter that adding a super pixel buffer can reduce the performance requirements for column level data transfer rate while maintaining the same readout efficiency as with higher rate and no additional buffering.

The front-end buffers were modelled using mailboxes in SV and FIFOs in SystemC. They both also have a TLM FIFO class (found inside an external library in SV) which implements many of the TLM interfaces. A pixel column was modelled as arrays of these buffers, one array for the pixels and one array for the super pixels. This idea can be extended to more than two levels of buffering if required. A counter per super pixel was used to model a transportation delay from a pixel to a super pixel buffer in a cycle-based simulation. Whenever a pixel has data available and the counter reaches zero, a datum will be moved from a pixel to a super pixel. In a timed simulation, a process per super pixel must be used for data transportation. These can be started automatically at the beginning of a simulation, or created dynamically on-request during the simulation.

3.2.5 Column bus and data fabrics

A column bus and its arbitration were modelled using a read pointer for the arbitration and a counter or a timed delay for the transportation delay. If a timed delay is used, one process per a column bus is required. The arbiter and the bus can be created as separate classes using TLM interfaces to make them more easily reusable. Another technique to model an ideal FIFO arbiter is to use an unbounded FIFO for storing hit addresses exactly in the order of arrival. Usually this kind of arbitration is very difficult to implement in the actual chip but offers a starting point for comparing different arbitration schemes. In this thesis, the FIFO arbitration was only used in the high-level simulations.

Data fabrics consisting of a number of data nodes, which were mentioned in the Chapter 2, were modelled using a for-loop which checks the status of each node. In a pull-configuration, each node that has buffer space available, checks the previous node for data. With this technique, the run-time is proportional to the number of nodes in the data fabric thus reducing the scalability for large fabrics with many nodes in them.

3.2.6 End-of-Column modeling

In a timed model, each EoC was modelled using a FIFO and one process. This process checks a column for data, blocks until data is available, writes data into the EoC FIFO, and then requests access to the periphery bus. A processing delay was also added to the process when a timed model was used. In a cycle-based model, as described in the algorithm at the beginning of this chapter, EoC is a passive block whose state is queried before writing data into it. Before reading from the cycle-based EoC model, the reader which is either output link or the periphery bus, queries if EoC has any data.

The periphery bus is modelled exactly in the same way as a column bus. Instead of pixels or super pixels, a number of EoC blocks will be connected to this bus. It is also possible to use a data fabric instead of a bus to transport data at the periphery as will be shown later. This can also be modelled in the same way as a data fabric in a column. If the column bus and the data fabric models are crafted carefully, they can be reused completely at the periphery.

3.2.7 Output link simulation

Output links and serialisers, which send the data off-chip, and can be running at several GHz, have to be carefully designed at the layout level and usually implemented in a full-custom manner. For the high-level models however, a behavioral description is enough to assess the performance of the architecture. In a timed model, each link was modelled as one process which has a fixed delay per data hit if all hits have a fixed packet length. A delay per bit was used in the simulations in which the packets didn't have a fixed length. From a data structure modelling a packet, the length of the packet was calculated at run-time.

Modelling the output bandwidth limitations of the chip is especially important for obtaining correct results if the output bandwidth is the limiting factor or bottleneck for the performance.

3.2.8 Simulation benchmarking

To evaluate the run-time of architectural models compared to corresponding RTL blocks, a set of benchmarks has been established. Impacts of testbenches and verification environments have been estimated to be less than 10% of the run-time using run-time profiling tools. The RTL simulations have been run using a commercial simulator, and the sequential high-level

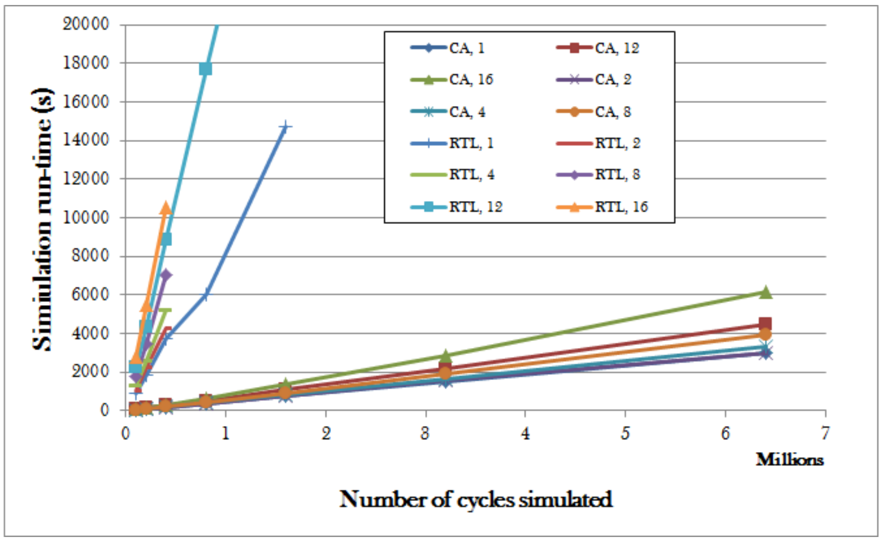


Figure 3.3: Run-time of RTL and sequential cycle-accurate (CA) simulations of the same model using different hit rates and different simulation lengths. The number after each label shows the relative input rate compared to other simulations.

models compiled using gcc. The simulations have also been run using one processor core only.

Figure 3.3 shows the run-times of these simulations for a readout ASIC VeloPix which will be presented in Chapter 6. The numbers from 1 to 16 after the labels RTL and CA show the relative input hit rate (data activity) of different simulations. The important thing to notice is that the run-time is a function of the number of cycles simulated and the hit rate per cycle. Not surprisingly, a high-level, sequential model done using C++ performs better than the SystemVerilog RTL model. Notice that a model whose run-time was proportional only to the activity, and not the length of the simulation, would be presented by horizontal lines in the plot. As there are always some background processes and monitoring required, if not from the model but the simulator kernel itself, it is difficult to create such a model.

The modelling and debugging efforts are difficult to estimate objectively because they depend on the expertise of the system architect and on the type of system being modelled. Complexity of a given block can be estimated from the LoC to some extent however. If an architectural model needs considerably more LoC than a corresponding RTL design, this acts as an indicator that the model may be too detailed or complex. For the models of Figure 3.3, the following LoC were obtained: RTL model 2670 and the sequential high-level C++ model 1329 LoC implying a factor of 2 reduction in LoC for the high-level model. No testbench code is included in these numbers. All other things being equal, increased number of LoC can result in more defects and increased effort for debugging.

For another pixel readout architecture design implemented during this thesis, the LoC for the RTL code was 6574, and the LoC for a TLM model of the same architecture 2955, again both figures excluding the LoC for testbenches. Over a factor of 2 reduction can also be seen in this example. This is a clear indication that high-level models can reduce the coding effort and the amount of debugging. For example, in [84] it has been found, that a LoC metric can predict the number of defects in the code. There are other metrics, such as Cyclomatic Complexity (CC) which is formed by observing all possible paths of execution in the code, to measure the complexity of source code. One difficulty in comparing the RTL and higher level code is, that the execution of the RTL code mimics parallel execution of hardware while a high-level model can be made fully sequential without complications of multi-process execution.

3.3 Latency

The latency of a system is evaluated to determine the dynamic range of a time stamp to unambiguously associate each event with a given time stamp. If the latency exceeds the dynamic range of the time stamp counter, this association cannot be done correctly. High-level simulations were used to track the latency of a hit in certain locations in the architecture. If a hit is modelled as a class, any amount of additional information for debugging and performance measurements can be added to the hit. This is shown in Listings 3.1. In addition to the actual time stamp for ToA and the hit address, a queue or an associative array of extra time stamps is included. Upon arrival of a hit, each component in the hit processing chain can then add a time stamp to the hit. In this way, bottlenecks for performance that cause the latency to increase can be more easily determined than in RTL simulation where high latency is observed only at the output of the chip. Determining intermediate latencies using signals and a waveform viewer from RTL simulation is slow and must be done manually. It is possible to add the latency information into RTL model, but this requires adding extra monitoring registers and buses for transporting this data between components. Each RTL component must also have access to the time stamp counter via a port to obtain the time information.

Listing 3.1: An example of a hit with additional timing information.

```
class HitPacket;
    int time_stamp;
    int address;
    int extra_time_stamps[$];       // Using a queue
    int extra_time_stamps[string]; // Or a map
endclass: HitPacket
```

Generally there are two components for the latency [85]. The first component is the time spent in the buffers by data, waiting for access to shared resources. It is usually variable and depends highly on the resource utilization or occupancy of the system. It is typical of this component to increase exponentially when the system utilization approaches 100%. However, sometimes it is also possible to guarantee an upper limit for the latency by design. In Chapter 2 it was shown that by choosing an appropriate arbitration algorithm for a data fabric, a latency with an upper limit was observed. The second component is communication or data transfer latency. This can be either fixed or variable. When a buffered data fabric is used for the data transfers in a system, the two components for the latency become intertwined because the data transfer will also have some waiting time.

3.4 Power consumption

Power consumption of a system cannot be solely determined from high-level models, but the models can be used as tools to estimate the power consumption of the full architecture. Once an architecture with sufficient performance in terms of efficiency and latency has been found, a super pixel or even a pixel block can be designed at RTL. This block can then be synthesized and a prototype of the physical design completed. A back-annotated netlist of this prototype can be used in simulation to obtain toggling rates for all nets in the design in the form of value-change-dump (VCD) information. Again back-annotating this information into a physical design tool, accurate estimate for power consumption of this block is obtained. The block can be characterized with different activity factors, for example idle 90% of the time and active 10% of the time.

A similar activity factor map can be obtained for a full design from the high-level simulation. Using the power estimation from a single block and the activity map from the full architecture, an estimation for the power consumption can be obtained. Thus taking advantage of the homogeneous structure of the pixel matrix and even the EoC blocks and the high-level models, characterizing only a few blocks can give an accurate estimate of the total power consumption. Because recording VCD files during a back-annotated simulation adds to the run-time overhead of simulation, a longer power profiles for a system can be obtained by combining the activity-based power information and the high-level simulation.

3.5 Simulation warm-up period

A warm-up period is a widely known problem in discrete-event simulations [85]. The same problem is also present in sequential, cycle-based simulations. The warm-up period means that when a simulation is started on an empty system (in an initial state), the behavior can be significantly different at the beginning than after a longer period of simulation, when the response of the system has become steadier. There can of course be systems which are constantly oscillating and never settle. In fact, the 'steady' state often means a convergence of the average of some measured value towards a range of values. In systems with non-deterministic components in them, for example economic systems or random arrivals of particles, a static steady state is never reached. These systems can exhibit dynamic behavior over some short period while the behavior may appear to be static or in a steady state over longer period.

Unless this warm-up period is determined, and its impact on the final results estimated, it may skew the results toward a certain direction. The results for queueing and buffered systems can be too optimistic because all buffers are empty at the beginning of the simulation, and thus have their full capacity available. Note that choosing too long a warm-up period is not detrimental to results but may increase the run-time of the simulation.

There are different ways to overcome the problem of the warm-up period, also known as the initial transient [86]:

- Running a simulation for long enough to make the impact of the warm-up period insignificant.
- Excluding the data from the warm-up period from the final data analysis.
- Choosing initial starting conditions that are close to the expected steady state.

In this thesis, the first method is adopted due to its simplicity. It requires no additional processing of data or manipulation of the system before the simulation. The drawback of run-time overhead is compensated for by using high-level behavioral models which run an order of magnitude faster than synthesizable RTL models.

If the warm-up period is analyzed in the context of HPD readout ASICs, issues described above can be found. Figure 3.4 demonstrates issues related to this warm-up period in one of the simulations performed during the work for this thesis. It shows cumulative moving average (CMA) of latency calculated after every 1000 packets (in red) and the average latency of the last 1000 consecutive packets (in green) in different points in time. It can be seen that at the beginning of simulation, the variation of CMA is larger from

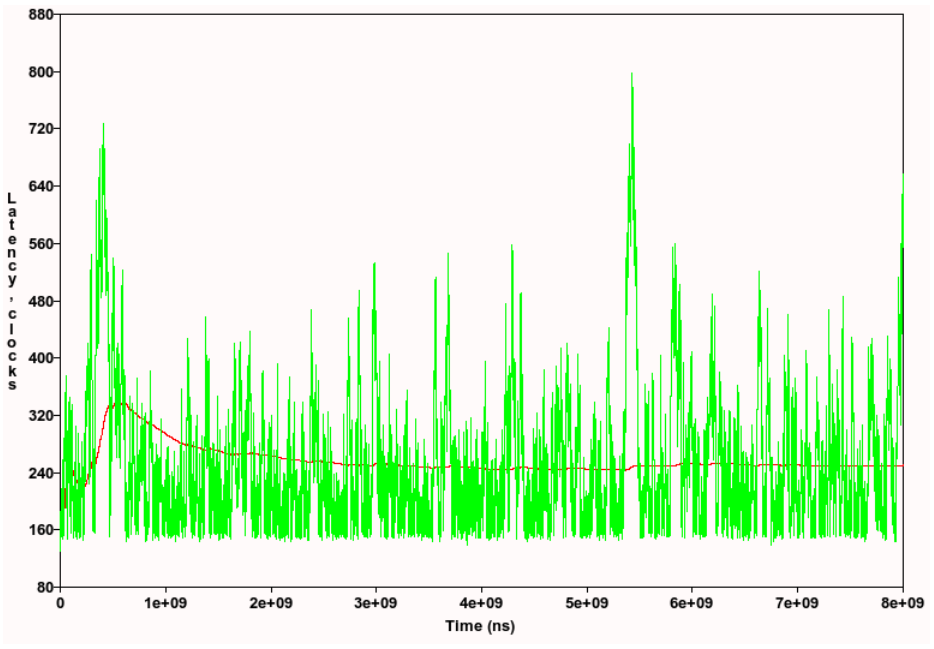


Figure 3.4: CMA of the latency (red) and the average latency of 1000 consecutive packets.

one sampling to the other. This can be considered as a warm-up period in which the system is started from an empty state. If simulation was stopped during that period, and the calculated latency taken as the final result, the steadier state and higher average latency would be missed. A steadier state here implies that the variable of interest (CMA of the latency in this case) is within a smaller range of values than in a less steady state. The other possibility would be to take too large a value as the final result if the CMA of the latency was sampled at the highest value of the red curve.

It can also be seen that the average latency of each sampling period of 1000 packets has a lot of variation throughout the simulation. This is not necessarily always the case for a random variable but is only shown here because the latency is relevant metrics for simulations in this thesis. It can be seen that at the beginning of the simulation, this latency differs from what is measured after the warm-up period.

Figure 3.5 shows the average latency plotted every 1000 packets for ten simulations runs. The system parameters (buffer size, readout rates etc.) are exactly the same, and only the seed number for the RNGs have been changed. This seed number controls the generation of hit arrivals and the address distribution of hits. In each run, 16 million hits were injected into the system. It can be seen that the values fluctuate rapidly at the beginning of the simulation before converging into narrower range of values, except

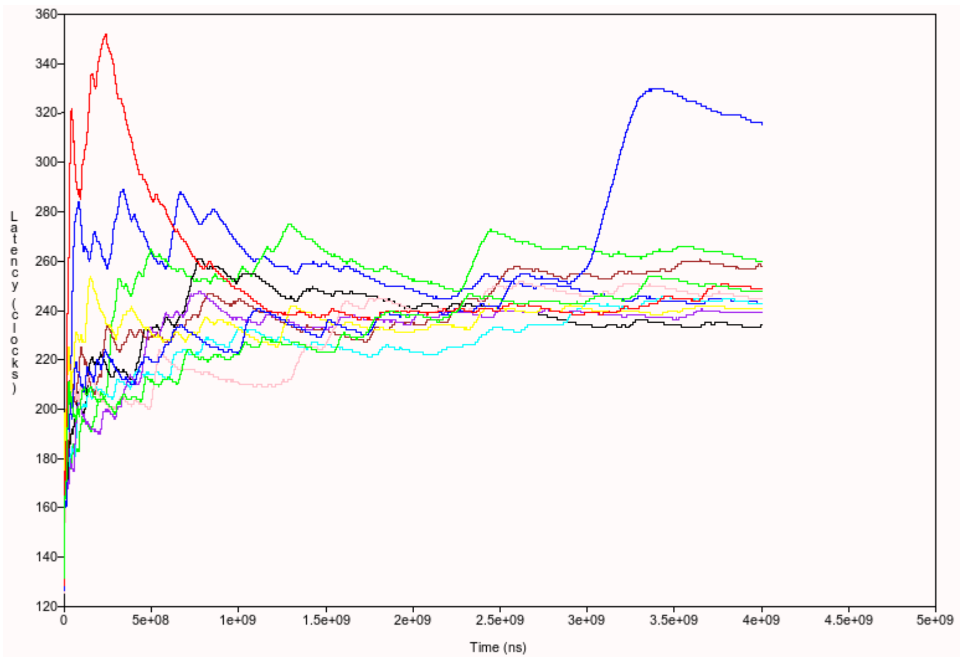


Figure 3.5: CMA of the latency in clock cycles sampled every 1000 packets versus time. Each colour represents a different seed number for the input data generator.

for one of the plots. It is interesting to see that this plot seems to stabilize between 240 and 260 clock cycles before shooting up. The root cause of this behavior was not discovered, but it can be attributed to either a smaller than the average times of arrival between hits or hit addresses that are close to each other creating local congestion, or both. These kinds of events are statistically more unlikely but not impossible, and should be accounted for in the design of the system.

3.6 Concluding remarks

In this chapter, simulation techniques used in this thesis were described. Some caveats and difficulties in performing full chip simulations for hybrid pixel readout chips were described, and some solutions were given. Two examples were presented in which the “higher than RTL” models were more concise and described in less LoC than the corresponding RTL models. It was seen that ”higher than RTL“ simulation and modelling techniques can speed up the architectural simulation of a pixel chip. Especially, in the presence of lengthy (in wall clock seconds) initial simulation transients, they can in fact be the only feasible way to study different architectural options.

As HPD readout ASICs start using newer CMOS technologies (65 nm and beyond) instead of the now widely used 130 nm technology, more complex architectures must be devised to meet the more stringent requirements for performance. Architectural simulations can be used as a tool guiding the design of these architectures to meet the specifications and managing the architectural complexity. However, they are not panacea for all issues brought by the complexity of these new technologies, and thus their deployment should always be considered for each use case separately.

Chapter 4

Hardware implementation studies of readout architectures

4.1 Background

A system-level analysis of different digital readout architectures and their data transfer characteristics of HPD ASICs was made in Chapter 2. In that chapter, metrics such as the readout efficiency and the latency were obtained using RTL and higher level C++ and TLM simulations. Chapter 3 discussed simulation and modelling techniques to be able to simulate large architectures while collecting sufficient amount of statistics for analysis of the architectural characteristics such as the latency and the readout efficiency. However, the full analysis requires taking into account physical implementation details such as area, power and timing. Because the pixel size is often dictated by the application, there is no option to increase the pixel size and the die area, and thus integrate more functionality at the cost of increased unit cost per die. While keeping the power consumption below a certain level may not be as strict a requirement as the pixel size in all applications, limited routing area available for the on-chip power supply imposes limits on the current that can be supplied to the pixel matrix. The timing performance, or the maximum frequency, may also be fixed by the timing resolution requirements of the application, and it is important to verify that the architecture meets the demands for throughput using this frequency. Thus, in this chapter, the architectures presented in Chapter 2 are analyzed in terms of hardware implementations, and their physical characteristics are compared.

All power, area and timing-related information has been extracted from a commercial 130 nm CMOS process. This process uses a stack of 8 metals

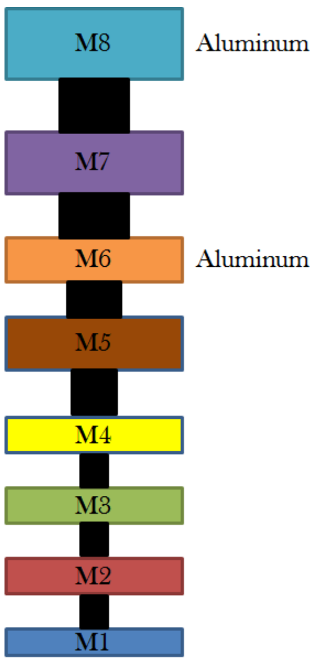


Figure 4.1: The metal stack of 8 layers of the CMOS technology used in this thesis.

having 1 bottom metal, 3 thin metal layers, 1 thick metal, 1 thin aluminum layer, 1 very thick metal layer and 1 very thick aluminum layer (see Figure 4.1). All layers are copper unless stated otherwise, and are referred to as M1-M8 in later discussion. Simulations presented in this chapter have been done using several process corners in different conditions. All delays reported are obtained using slow process corner (SS), temperature of 125 C° and a power supply of 1.4 V. All dynamic power consumption figures reported have been obtained using fast process corner (FF), temperature of -55 C° and a power supply of 1.6 V. These values were used because the commercial standard cell library was characterized by the vendor using these conditions. Unless explicitly stated otherwise, these conditions hold for all simulations.

Two 130 nm standard cell libraries were used for these estimations, a commercial library and a customized standard cell library. The biggest trade-off between the libraries is that the customized library is optimized for area ($\times 2$ reduction) by reducing the driving strength of the cells, and also optimized for leakage reduction using high V_t transistors. The commercial library has more cells than the customized library (a factor of 10) but the synthesis results have shown the cell count increasing by less than 5 % when using the customized library. Leakage reduction of several orders of magnitude has been estimated by the digital library characterization tools at room temperature or higher.

Gate-count is typically used in the literature to estimate the hardware cost of a specific architecture. Because in this thesis the absolute size of the pixel is a very important characteristic, all area estimates are expressed in μm^2. If desired, the equivalent NAND-gate count can be obtained by dividing this area with the area taken by one 2-input NAND-gate.

Power consumption is divided into static and dynamic power consumption. In this chapter, the main interest is on the dynamic power consumption, which can be expressed as

$$P_{dyn} = \alpha f C V^2 \tag{4.1}$$

where α equals the activity factor or the toggling rate, f the clock frequency, C the total switching capacitance and V the power supply voltage. The difficulty in applying (4.1) is that when the clock gating is used in many clock paths, α will not be uniform across the full system, and it can be difficult to estimate how much capacitance is switching at which activity rate in any given moment. Also, if (4.1) is directly and homogeneously applied across all capacitance with a constant α, this can lead to a pessimistic estimation. Thus obtaining the power consumption using a synthesized netlist and a VCD file from post-synthesis or post-layout simulations using an expected stimulus of the application can give a more accurate prediction of the dynamic power consumption.

To evaluate and compare different architectures, two metrics are used. To be able to compare heterogeneous architectures, power consumption must be evaluated as effective power $P_{eff} = mW/Mhits/s$. P_{eff} is then a measure of how much power must be expended to achieve a certain output rate. Similarly, the area footprint is evaluated as $A_{eff} = \mu m^2/Mhits/s$ stating how much area must be used to achieve a certain hit rate. In both the cases, the unit Mbps can be substituted for Mhits/s if the number of bits per hit is known. These are means to evaluate the architectural efficiency in terms of area and power. All applications also have total area A_{tot} and power P_{tot} requirements. Thus, P_{eff} and A_{eff} cannot be optimized without taking these constraints also into account. Also, applications usually have a minimum output rate R_O requirement which must be taken into account when choosing the architecture.

4.2 Digital pixel implementation

In the following discussion, the digital pixel logic is separated from the readout logic. This separation is difficult because the readout logic and the measurement logic (counters) can be tightly coupled. This is especially true for serial shift register implementations [43, 42, 25, 35]. Also in [75], digital pixels compute their ToT values individually but the ToA value is handled by a shared buffer. On the other hand, functionality such as synchronization needs to be always done for each discriminator input separately, and thus is considered part of the individual pixel logic.

In the past, many pixel chips have incorporated full custom digital logic into the pixel [43, 25, 87]. However, using an area-optimized standard cell library and digital design tools, similar density has been achieved using a synthesizable Verilog description [42]. This is also the approach taken in this thesis, and makes the results more easily reproducible because layouts are generated mainly using scripts. Porting the architectures to newer CMOS technologies will be more straightforward because the same Verilog code and many of the implementation scripts can be reused. Note that the development of an area-optimized standard cell library is still required when moving to new technology, if an increase in transistor-density proportional to the scaling is desired. Because the area-optimized library increases the density by 2 compared to a non-area optimized library, without the development of a new library, only a factor of two increase in density is achieved.

In [88] a comparison between on-pixel counters implemented using a linear-feedback shift register (LFSR) and configurable counters is shown. Two 15-bit LFSRs can be implemented using 530 transistors and the configurable depth counters using 626 transistors. Configurable counters support either two counters of 14 bits (or smaller) or one counter of 24 bits. Thus, with an extra overhead of 18 % in area, more general purpose approach is reached. Using LFSRs instead of binary counters can give an area reduction of 30 % [89]. The LFSR also supports shifting of data out of the pixel by adding one extra multiplexer, while in a binary counter a multiplexer for each flip-flop must be added.

One option utilized in the pixels for example in [46] is to latch time stamps into parallel load registers on a leading and trailing edges of the discriminator signal. This solution does not require any extra gates for the flip-flops but does not support shifting of the data out of the pixel. On the other hand, if there are enough routing resources available for full parallel readout, this option is very useful for achieving a high throughput and minimising gate area inside the pixel. This kind of parallel time stamp latching can also be utilized with static random access memory (SRAM)-based architectures [53].

If a counter is required instead of a parallel load register, an asynchronous binary ripple counter can also be used to minimize the number of extra gates. This counter does not support shifting without adding extra gates, and requires also clock multiplexing in this case. Another feature of this counter is that because external clock signal is connected to only one of the flip-flops, it is robust against timing errors. If a clock pulse does not trigger the flip-flop because of synchronization error, it does not corrupt the value of the counter. This corruption of data can happen in a fully synchronous counter in which the clock signal is sent to all the flip-flops if some flip-flops treat the clock signal as valid and some treat is as invalid. An example of this is for example, when a binary counter holds value 4'b1011, and would transition to state 4'b1100 when incremented by one. If only the first flip-flop observes the clock pulse, the state will incorrectly be 4'b1010, so the counter has been actually decremented by one.

Generally, asynchronous finite state machines (AFSMs) have been deployed in the digital front-ends for synchronization instead of flip-flops and synchronous FSM [25, 35, 75].

4.3 Architectures using shift registers

Although it was shown in Chapter 2 by simulation that shift register -based architectures require large output to input rate -ratios to achieve readout efficiencies of 99 % or higher, sometimes their usage is required due to very restricted area available. Especially efficient implementation in terms of absolute area is reached if it is possible to use the same flip-flops for shifting that are used for counting. However, this is not always the case and there are architectures which have dedicated flip-flops just for shifting [47].

Serial shift register architectures are used in [43, 42, 25, 35]. The drawback in serial shift register architectures is, that all registers forming the shift register need to be clocked during the shifting, unless some kind of zero suppression technique is used such as in [35]. Controlling the shift register is simple in this case, and done using a multiplexer and a global $shift\_enable$ signal. A state machine is not required per pixel to control the readout operation.

In [12], a "conveyor belt" architecture is presented. This is a parallel shift register based architecture in which only address information of pixels (6 bits) is stored in the register. The time stamp is derived from the fact that the latency to shift the bits from a specific pixel is always fixed, and proportional to the row address. For example, shifting address from a row 10 takes 10 clock cycles. It also includes time stamp correction bits in the case that the hit cannot be written into the register at the moment of the hit. The advantage of parallel architectures over serial ones is that they have more throughput and the clock gating can be controlled for each parallel register. The disadvantage is that without adding extra logic, the same flip-flops cannot function as a counter and a shift register. Routing overhead is also increased because multiple bits need to be connected between consecutive registers.

4.4 Column bus architectures

A shared data bus is a commonly used architecture for transferring data between different modules on-chip. Typical data bus configurations are a tri-state bus, a multiplexer-based and AND-OR-based bus [90]. Each bus has a number of masters who initiate bus transactions and slaves who respond to these transactions. A global on-chip bus can consist of wires which span almost the full size of the chip in one dimension.

In pixel readout chips the buses are typically routed from the top of the columns to the bottom [53, 54, 49, 37, 46]. Because the pixel matrix occupies most of the chip area, these buses cover almost the full distance of the chip. On the other hand, these buses are uni-directional having multiple transmitters (TXs) (pixels or super pixels) and one receiver (RX) usually located at the End of Column (EoC). Figure 4.2 shows two different implementations of a uni-directional pixel column bus. Each TX is connected with a wire of length L and width W. The length of the connection from a TX to the bus trunk L_{conn} is assumed to be negligible. The arbitration signals are omitted from the figure for clarity. Owing to the large aspect ratio of a column, it can be difficult to create a tree of connections to reduce the propagation delay through the gates (Figure 4.2a) because this requires more routing resources. Using a tri-state bus (Figure 4.2b) this delay can be avoided but the capacitance of the bus wires increases linearly in proportion to the wire length.

In Figure 4.2, a single piece of wire with length L and width W (capacitance C_w, resistance R_w) is modelled as a 3 π ladder circuit with 3 equal resistances $\frac{R_w}{3}$ and 2 different capacitance ($2 \times \frac{C_w}{6} + 2 \times \frac{C_w}{3}$) values. This model is more realistic than a one-step L model with one lumped resistance and one capacitance. The lumped RC-model has been found to be a poor approximation for on-chip wire delays while the 3 π model shows only a maximum error of 3% compared to a fully distributed RC-line [91]. The relative error of an L ladder model can be up to 30% [91]. Note that while the 3 π circuit model is preferable to a one-step L ladder in this case for accuracy, in a circuit with hundreds or thousands of RC-wires it may give rise to unacceptable simulation or analysis run-time overhead.

The wire delays in CMOS technologies are well-understood. In [92], for a 0.1 μm technology, delays of 1.2 ns were estimated for wires up to 5 mm. In [93], for a 5 mm bi-directional (tristate) bus, a delay of approximately 2 ns was estimated while transition times can be as high as 5 ns without even reaching 100 % of the signal level. Because the signal delay of the wire increases in proportion to the length squared, repeaters must be inserted to reduce this delay. One issue in the design of pixel columns is that thicker, lower resistance top-level metals are often reserved for power distribution to minimise voltage drops and ensure correct functioning of the logic. This

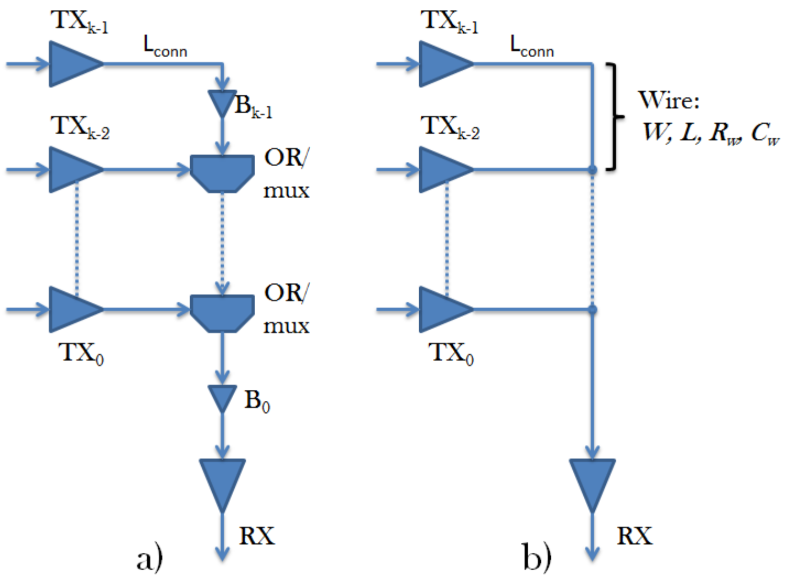


Figure 4.2: Schematics of two uni-directional buses: a) OR/mux-based b) tri-state bus.

limits the possibilities to use these higher level metals with lower resistance for routing of the signals. In this section, a bus length of 256×55 μm = 14.4 mm is considered. This is the column length in the readout chips presented later in this thesis. Simulation results from 130 nm CMOS using 0.2-μm wide metal shown in Figure 4.3 indicate that the delay can be up to 6.3 ns for a 14-mm wire. This was measured from 50 % of the input voltage to reaching 50 % of the maximum output voltage. By using repeaters inserted every 880 μm, this delay can be brought down to 1.6 ns. This number was used in one of the designs in this thesis, and was verified later from measurements from a manufactured chip. The size of the buffer required to achieve this delay was 4.8 μm $\times$ 14.8 μm with 20 fingers for NMOS (W = 570 nm) and PMOS (W = 1.8 μm).

To understand the difficulties of using a bus-based structure for transferring data from pixels to EoC blocks and the justification for using a data node-based fabric which introduces latency to the output data, the signal propagation delays were studied. The effective output rate R_O (data packets/s) or the data bandwidth versus the output latency is an important trade-off in this case. Given the same output rate R_O for a bus-based and a node-based architecture of equal area and power, a solution with higher efficiency and lower latency is preferable. There are other considerations such as noise and digital-to-analog coupling which are not taken into account. However, it will be seen that the bus-based solution introduces significant

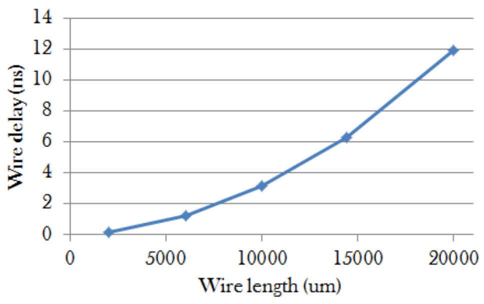


Figure 4.3: Simulated wire delays with wires from 2000 μm to 20000 μm.

Table 4.1: Transistor widths in output inverters of 2-input OR-gates (L = 120 nm).

Type	W (PMOS)	W (NMOS)
B	520 nm	330 nm
D	1.44 μm	860 nm
E	1.97 μm	1.13 μm
K	6 $\times$1.64 μm	6 $\times$1.03 μm

load capacitances and hence needs larger buffers to have a performance equal to the node-based architecture. The large transient currents through these buffers can increase the digital power supply noise. This noise may then couple to the analog signals or to the analog power supply. Using larger buffers also increases the power consumption as will be shown later in this chapter.

A column length of 256 pixels of 55 μm is assumed because the architectures under study in this thesis have that dimension. The column width is assumed to be 2 $\times$ 55 μm because the double column architecture is used. It is also assumed that there is one TX per 4 pixels, and a total of 64 TXs per (double) column. Taking the column width into consideration is not crucial for the timing of the column bus but has an impact on the area available for buffering of the bus. Figure 4.4 shows the worst case delay of a 64 OR-gate column bus (see Figure 4.2) and the average power expended per bit using OR-gates of different driving strengths. The clock frequency used is 40 MHz. The sizes of the output inverters in each type of OR-gate are listed in Table 4.1. P_{eff} is lowest with the type E, being 0.85 μW/bit/MHz. The type B has the worst P_{eff} = 1.24 μW/bit/MHz. Using these numbers, a comparison to other architectures can be made by scaling the number with the number of bits in one hit event.

As was discussed in Chapter 2, an arbitration scheme of the column bus can be implemented using a priority encoder or a token ring. Schemes such

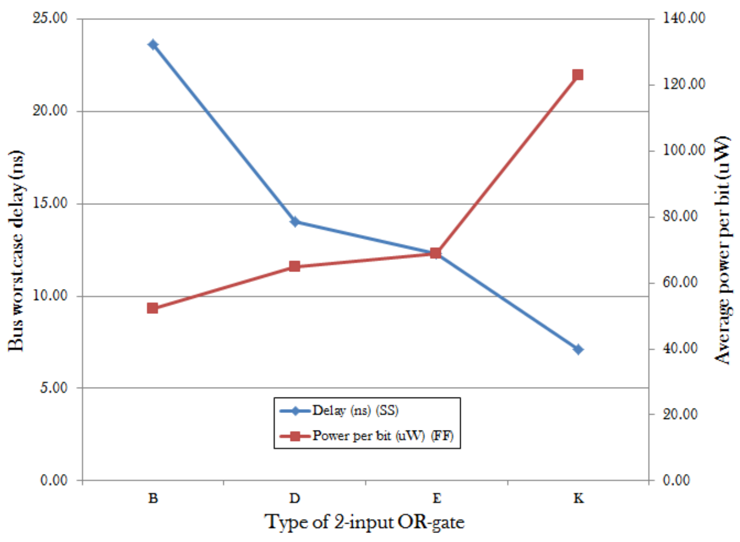


Figure 4.4: The worst case delay (slow corner) and average power per bit for a column bus (fast corner).

as centralized arbitration are difficult to implement inside a column because point-to-point connections cannot usually be made between the arbiter and each requesting block due to the limited routing area available. A token scheme for a column bus using alternating NAND- and NOR-gates between pixels to minimize the propagation delay is presented in [94]. This kind of structure is faster in terms of propagation delay than a structure containing AND- or OR-gates only. For 64 modules connected to a token ring, a delay of 20.7 ns was obtained from simulation (slow corner) using alternating NAND- and NOR-gates. NAND- and NOR-gates with the weakest driving strengths (type B) were used.

For a fully synchronous system, the token must either reach any module in the ring in one clock if there are no subsections in the ring. It was shown in Chapter 2 how the ring can be divided into subsections to relax the timing constraints at the cost of extra arbitration latency, and generally each subsection introduces an extra clock cycle of latency and a corresponding decrease of the throughput. If the subsections are used, this also means that the token cells at the higher level of the ring must be connected with longer wires to bypass all modules at the lower level of hierarchy in the ring.

4.5 Clock distribution in a column

The clock signal generally serves two different purposes inside a pixel column. First, it must provide an accurate, global timing reference across the full chip for timing measurements. Second, it is used to clock all the synchronous logic within the column.

Typical requirements for clock distribution are [95]:

- Low skew and low jitter.
- Fast signal rise and fall times.
- Low distribution delay.
- Tolerance to process, voltage and temperature (PVT) and on-chip variation (OCV), and meets all requirements in all simulation corners.

The first property is especially important if accurate timing measurements need to be performed. The problem of skew in the clock distribution comes from the fact that the clock needs to be distributed over long distances of $> 1cm$ in large pixel chips. Due to the large aspect ratios of pixel columns (100-to-1 or even larger), vertical routing resources are scarce. This means building a balanced clock tree by placing the first buffer in the middle of a column is more difficult. Because low-resistivity top metal layers (for example M7 and M8 in the 130 nm technology studied here) are usually reserved for power distribution, these layers are not available for clock distribution. Delay lines can be used to reduce the skew at the cost of extra area for the cells [75]. The problem is that the lines need to be heterogeneous for different areas of the column, being longer at the bottom and shorter at the top of the column. Another option is to include programmable delay lines for tuning the clock skew. However, this option requires extra configuration registers and a DAC for controlling the speed of the delay line.

In [25], the problem of skew is neglected altogether, and each pixel has an inverter driving the clock to the next pixel. This causes variation of the clock skew up to the full period of the clock signal. This delay must then be taken into account in off-line analysis.

Fast rise times can be ensured by inserting repeaters in the clock tree instead of driving long sections which will slow down the edges. In this thesis, all architectures presented here deploy a similar clock tree architecture. This reduces the bottom-to-top skew to less than 2 ns even in the slow process Corner. For the 130 nm technology, wire width of 0.4 μm was required to reduce resistance of the line to meet this skew target. By using a similar clock tree in all the architectures, the power consumed by the clock distribution can also be more easily separated from the power consumed by the readout logic. This clock tree architecture is shown in Figure 4.5.

This clock tree is divided into sections of 880 μm and the trunk is buffered with a relatively large buffer (CLK_Q) at the same intervals. This buffer uses regular V_t transistors to reduce the delay. The width of the transistors used were already given earlier in this chapter when discussing wire delays.

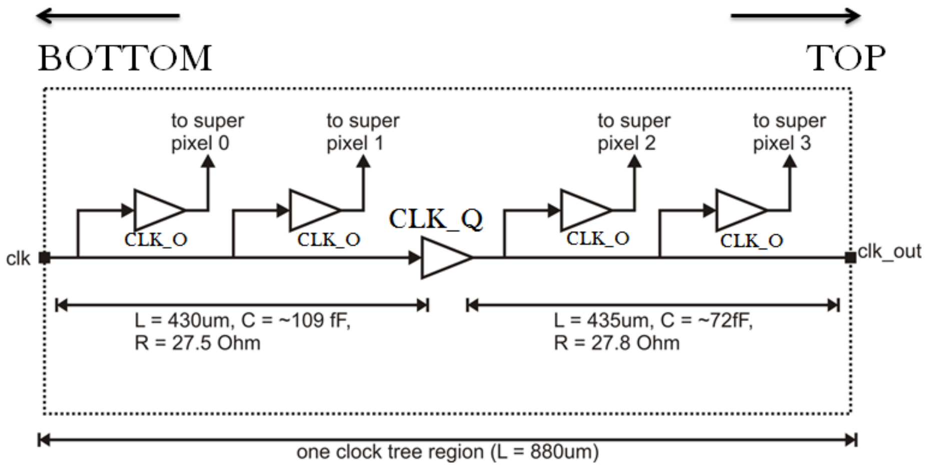


Figure 4.5: The clock tree of a pixel column consisting of repeaters connected in series.

The smaller buffers (CLK_O) are used to buffer the clock to each super pixel or pixel regions. No other connections to the clock trunk are allowed to limit the capacitance of the trunk. This restriction can be achieved by using set_dont_touch command. Note that the left section of the clock tree has higher capacitance due to the relatively large input capacitance of the CLK_Q buffer.

When relying heavily on clock-gating techniques for dynamic power reduction, the insertion delay in the clock tree can become a problem. This can create hold violations at the bottom of the column in the first pixels or super pixels if the delays increase beyond a few nanoseconds. In this thesis, this problem was solved by clocking the signals sent from the EoC to the column at the negative edge of the clock. The drawback of this approach is that it effectively cuts the available clock period in half for meeting the timing requirements for global signals which must travel the full span of the column.

4.6 FIFO implementations for storing pixel data

As was shown in Chapter 2, data (memory) buffers are needed in the simulated readout architectures. Buffers have been sparingly used in existing pixel readout architectures to provide multi-hit capacity and to reduce losses related to the digital front-end pile-up [75]. Their selection must be analyzed in terms of area and power consumption. One important consideration is the choice of FIFOs. A comparison of different FIFO types is presented in [96]. It lists three main types for FIFOs:

- Shift register based FIFOs.
- Exclusive read/write FIFOs.
- Concurrent read/write FIFOs.

The first option is not particularly useful for storing data while waiting for access to a shared resource. Read- and write-operations must be always performed at the same time into this kind of structure. The focus in this chapter is on the second type, where reads and writes are synchronous with the same clock, and may happen at the same time or during different clock cycles. In the third option, the operations can be completely asynchronous of each other. This option rarely needs to be used in the pixel matrix because of the relatively restricted area using one clock domain. In fact, there are no existing implementations using asynchronous FIFOs in the pixel matrix.

One particular feature of the FIFOs in the pixel matrix is that they need to be small (typically less than 8 words) due to the restricted area available. This means that SRAM-based solutions are not feasible due to overhead caused by sense amplifiers and address decoding circuitry. Another consideration is whether a FIFO has to be SEU tolerant or not. This can effectively triple the amount of control logic required, thus increasing the area footprint considerably. Finally, data stored in the FIFO can also be protected. This can be done by encoding the data internally, and exposing only decoded data to modules outside the FIFO. If data is externally encoded using ECC, and the FIFO is used only for storing the encoded data, it does not change the internal structure of a FIFO.

Two typical FIFO architectures are a linear FIFO and a FIFO based on pointer logic. The first one requires N control bits and the second one $log_2(N)$ control bits, where N is the maximum capacity of the FIFO. The first option has longer latency because each word must fall through all memory slots while in the second option the next word is available on the following clock cycle after write-operation, and in the case of the FIFO having more than one word, also after a read-operation. In this case, latency means explicitly the number of clock cycles, and not the absolute delays in the logic. While a linear FIFO can be clocked at higher frequency, the clock frequency in pixel readout chips is usually fixed by the application, and frequencies up to 100 MHz are used [25, 49, 75].

4.7 Data fabric implementation

Although data fabrics could be considered as parallel shift register architectures, the fabrics are treated separately from shift registers here. This is due to the fact that the control of the fabric is completely decentralized and based on local handshaking, and there is no global *enable* signal. The arbitration is also more complex but also allows some flexibility in the implementation.

The functionality of the node-based fabric and the data network on-pixel chip were already analyzed and their performance assessed in Chapter 2. The goal of this section is to examine the physical considerations when implementing one of these architectures. Because a number of additional memory or flip-flops are required in each node of these architectures for storing the data packets, this may increase the power consumption and area footprint considerably. For example, in [97] a power consumption of over 25 μW for a flip-flop implemented in 130 nm CMOS running at 1 GHz has been obtained. During the work for this thesis, approximately 1 μW at 40 MHz for a flip-flop of the same technology was obtained when the flip-flop was clocked but the stored data did not change. For a matrix of 128×64 nodes this equals to over 200 mW of power when each node has a 24-bit register. Adding these registers reduces also the area available for counters in pixels. Thus, it may be necessary to share one register between multiple pixels to reduce the area overhead.

Due to the large parameter space and limited amount of time for the investigation, in the detailed implementation studies for the data fabric only the WRR arbitration scheme was used. This is also the simplest implementation because it requires only a binary counter which is then compared to a hard-wired address. Other implementations require either comparing the time stamps or adding additional registers for storing the waiting time of the data.

The hierarchical data fabric has a slightly different area and power footprint compared to the linear or one dimensional data fabric. For any given 2-dimensional hierarchical data fabric with N nodes, there exists an optimal group size K which reduces the number of operations needed to transfer a packet out of the fabric. When the number of these operations is minimized, also the data-related activity, and thus the activity factor α, within the fabric is reduced. The optimal groupings in terms of latency were already given in Chapter 2. Note that when $K = N$, the two-layer fabric is reduced to one layer again.

From the implementation perspective, increasing the group size K while keeping $K < N$ increases the difficulty of meeting the timing in the outer layer of the data fabric because the node distance is increased. The area per super pixel taken by the outer node decreases when K is increased, and also the number of memory bits required in each node is decreased. The reason for this reduction is that the arbitration counter can be made smaller, and the inner data nodes must only store the address related to the inner node fabric instead of the full fabric address.

More than two dimensions for the data fabric were not considered in these studies, because increasing the number of dimensions also increases the number of parallel wires required in a given area. Each dimension in the fabric requires its own routing resources, and using a fabric with three levels instead of one requires 200 % more routing resources. One option to reduce the routing overhead would be to split data packets into multiple words at the cost of reduced throughput. If the throughput can be reduced but the latency needs to be minimized, multiple transfers can sometimes be afforded. As an example, consider a linear fabric with $N = 64$ data nodes. The latency in this case can be up to 64 clock cycles for packets from the last node, not taking into account the waiting times in any intermediate buffers. It was already seen that this kind of fabric offers a throughput which is equal to $f/3$, which means that handshaking consumes 2 cycles and the actual data transfer only one. If the packet was split into two transfers, throughput would be reduced to $f/4$. But now the number of wires needed for routing would be half of the original. Then consider selecting a group size $K = 4$. This would make the outer layer in a two dimensional fabric to have 16 data nodes. Then the maximum latency through this fabric would be $4 + 16 \times 2 = 36$ clock cycles, not taking into account the waiting times in buffers.

4.7.1 Area and power consumption

Due to additional routing required by the hierarchical fabric, a column width of 33.6 μm was needed. This means that when assuming a pixel pitch of 55 μm, and a double column architecture, 76.4 μm is left for the pixel front-ends on both sides. For the linear fabric, a column width of 28.6 μm was enough to successfully route the design. The pin-level interface between two groups or a group and the EoC when using the linear fabric is shown in Figure 4.6. A 23-bit data bus between the data nodes can be routed, and a 9-bit time stamp can be supported. No effort was made to extend these numbers because this study was done for one of the applications presented later in this thesis, and these ranges were enough for the demands of that application. The clock tree was routed using a wire of twice the minimum width to decrease the resistance and the bottom-to-top clock skew in the column.

Figure 4.7 shows the dynamic power consumption of the linear and the hierarchical data fabrics when $N = 64, K = 4$ as a function of the data activity. The power of the clock and the time stamp distribution have been excluded from these figures. The clock distribution consumes 120 μW per group and the time stamp distribution approximately 10 μW per group. For a column with $N = 64, K = 4$ this adds up to 1.92 mW and 0.16 mW respectively. It can be seen from Figure 4.7 that group 0 has always the highest power as expected because all data must pass through these groups.

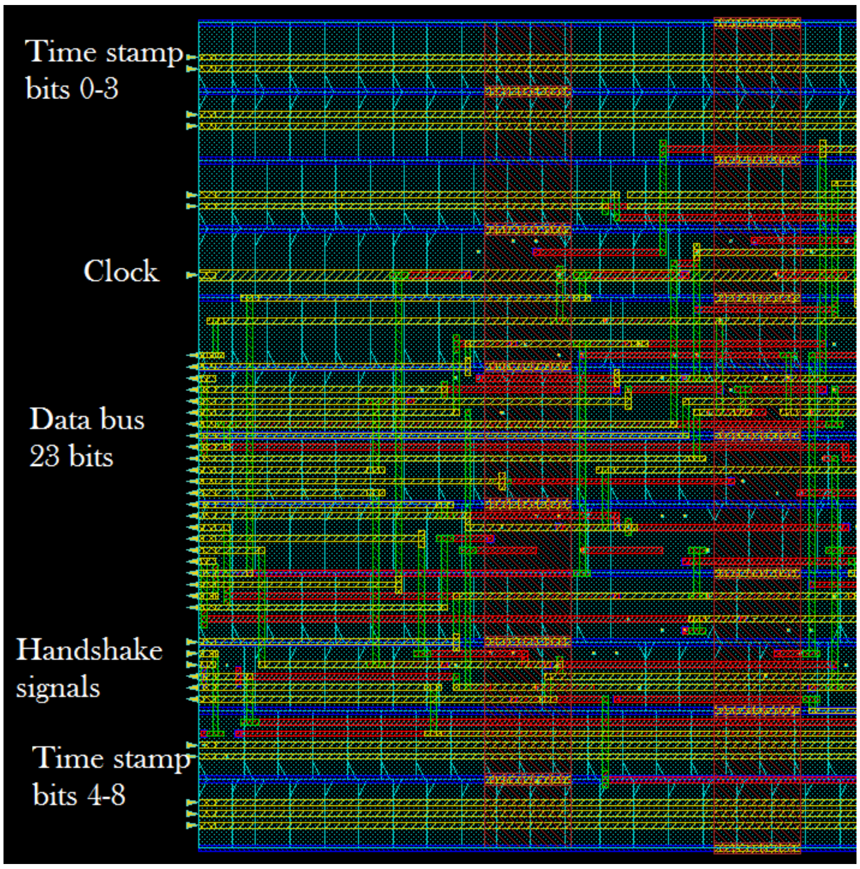


Figure 4.6: The pin interface between two groups in the linear fabric.

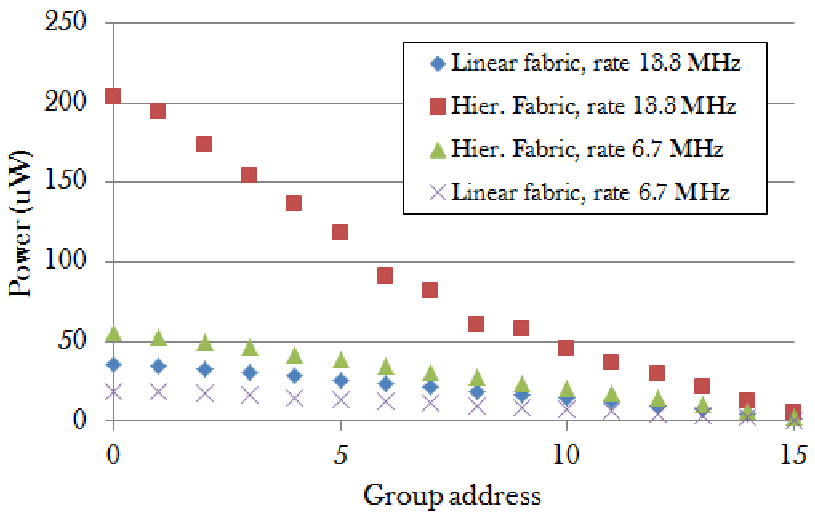


Figure 4.7: Data-activity related power consumption of the two fabrics.

Extracting the corresponding effective power for both the fabrics, the following numbers are obtained for the linear fabric: $P_{eff}(6.7Mhits/s) = 0.25$ mW/MHz and $P_{eff}(13.3Mhits/s) = 0.24$ mW/Mhits/s. For the hierarchical fabric, the numbers are $P_{eff}(6.7Mhits/s) = 0.70$ mW/Mhits/s and $P_{eff}(13.3MHz) = 1.4$ mW/Mhits/s. One of the main reasons for much higher power in the hierarchical fabric is that the node arbiters at the second level fabric need to arbitrate constantly between the local buffer and the previous node. This happens because it is four times more likely that there is data in the local buffer than when using a linear fabric.

If the contribution of clock and time stamping are omitted for comparison purposes to the bus-based architecture and only the data-related power consumption taken into account, the following figure is obtained for the linear fabric: $P_{eff}(13.3MHz) = 1.02$ μW/bit/MHz. This number is slightly higher than for an OR-gate based column bus using E-type drivers for a number of reasons. The arbitration is not included in the bus-based model. The control signals required for the bus are also omitted.

4.7.2 Maximum clock frequency

The maximum clock frequency of the data fabrics was investigated using a synthesis tool. No full PnR was done for this purpose but the wire loads were specified for the data input pins using the Synopsys design constraint (SDC) command set_load. The distance between the data nodes was assumed to be 220 μm, and this was obtained assuming 256 $\times$ 55 μm pixel, and one node per 4 pixels. The load used was 50 fF/220 μm. The slow simulation

Table 4.2: The synthesis results for the linear and hierarchical data fabrics.

Architecture	Clock period (ns)	Area / node (μm^2)
Linear fabric	6.65	3680
Hierarchical fabric, K = 4	6.90	3730
Hierarchical fabric, K = 8	7.00	3594

corner was used for this study. Several group sizes $K = 4, 8$ were used for the hierarchical fabric.

The results of the synthesis are summarized in Table 4.2. The table contains names of the architectures, the maximum clock frequency and the obtained cell area per one node. There is a difference of 0.25 ns in the minimum clock period between the linear and hierarchical data fabrics when $K = 4$. This is expected because there is an additional multiplexer required to select between the current group of K nodes and the previous nodes. When $K = 8$, the clock period increased by 0.1 ns only. Although increasing K increases the wire length between the nodes in the second level, as shown Figure 4.3, the wires delays for wire lengths below 2 mm are relatively small. It can be seen that the cell area per node is slightly decreased when increasing the K. This is due to the reduced number of memory bits per node within a group. While the second level node must store the full address always, the first level nodes within the group need only store the local address.

The maximum clock frequency can also be limited by the power budget of the application. Even though the effective power does not increase with the higher clock frequency, the absolute power consumption scales up with the increased clock frequency according to (4.1). Another limiting factor that was observed are timing constraints of global signals such as *readout_enable* , the shutter or time stamping distribution, if such signals are used. These are not specific or required features of the node-based architectures but are also used in other types of architectures.

4.8 Network implementation

It was shown in Chapter 2 that a network has certain advantages over one-directional data fabric, such as better tolerance to manufacturing errors and to noisy areas generating additional traffic and hotspots in the pixel matrix. The readout efficiency was also slightly higher for the same output rate when using a network instead of a fabric. However, the network has extra requirements in terms of implementation because digital communication is also required between rows of digital logic. This implies routing digital signals between or over analog regions causing potential coupling problems from digital to analog domain.

The network implementation presented here is targeted for pixel chips with $55\ \mu m \times\ 55\ \mu m$ pixels, also assuming a super pixel of 2×4 pixels, but other super pixel dimensions, especially larger ones, are also feasible. It utilizes a handshaking pull-architecture between two data nodes where the communication is done via $data\_valid$ and $data\_read$ -signals. Only 8 bits are transferred at a time to minimize the utilization of the routing resources.

Figure 4.8 shows a layout of a 4×4 network of 16 data nodes connected to each other. The upper right part of the figure shows the connections between four data nodes. The area reserved for analog front-ends is marked with red rectangles, each measuring 38.4 μm $\times$ 22.0 μm. This estimate is based on the analog front-end implemented in the same technology [16]. The area for digital logic is 64.8 μm, and this area must also accommodate the digital front-end logic. The total cell area for each node is estimated to be approximately 2400 $\mu\mathrm{m}^2$ by the synthesis tool. This means that the area taken by the readout logic from the total area available for the digital logic is 16 % if a super pixel of $2\ \times\ 4$ pixels is assumed. There are also digital routing channels between two double columns for horizontal data connections. Each channel is 16.6 μm wide. Having multiple channels per super pixel is not an optimal solution in terms of the layout area because the routing occupancy is very low in the channels. In this network implementation, $2\ \times$ 10 bits are required horizontally between the data nodes for data routing, and additionally 4 bits are needed for the communication. With a routing pitch of 0.4 μm, the most compact implementation could use one channel of less than 10 μm.

Power-wise the digital logic in the network has higher power consumption than the one-dimensional fabric. To accurately estimate this difference, a full placement and routing of the pixel matrix would be needed. Because neither of the two application presented later in this thesis uses the network, this step was not done. However, because the network is effectively a more complex implementation of the one-dimensional fabric, remarks about the power consumption can be made.

Similarly to the fabric, each node in the network contains a register for storing one packet at time. Data must be propagated from this register to all possible data sinks increasing the power dissipated in wires compared to the one-dimensional fabric. Because arbiters at the input and output of the network nodes are more complex and need to arbitrate over several clock cycles, this increases the power consumption. Each network node also has a 4-to-1 input multiplexer for data packets instead of a 2-to-1 multiplexer. This increases the area requirements and adds to the extra power consumption as well. Also, theoretically it is possible to reduce the power consumed by clock trees by using the same clock tree in multiple columns of nodes because the clock signal could be routed between the analog regions

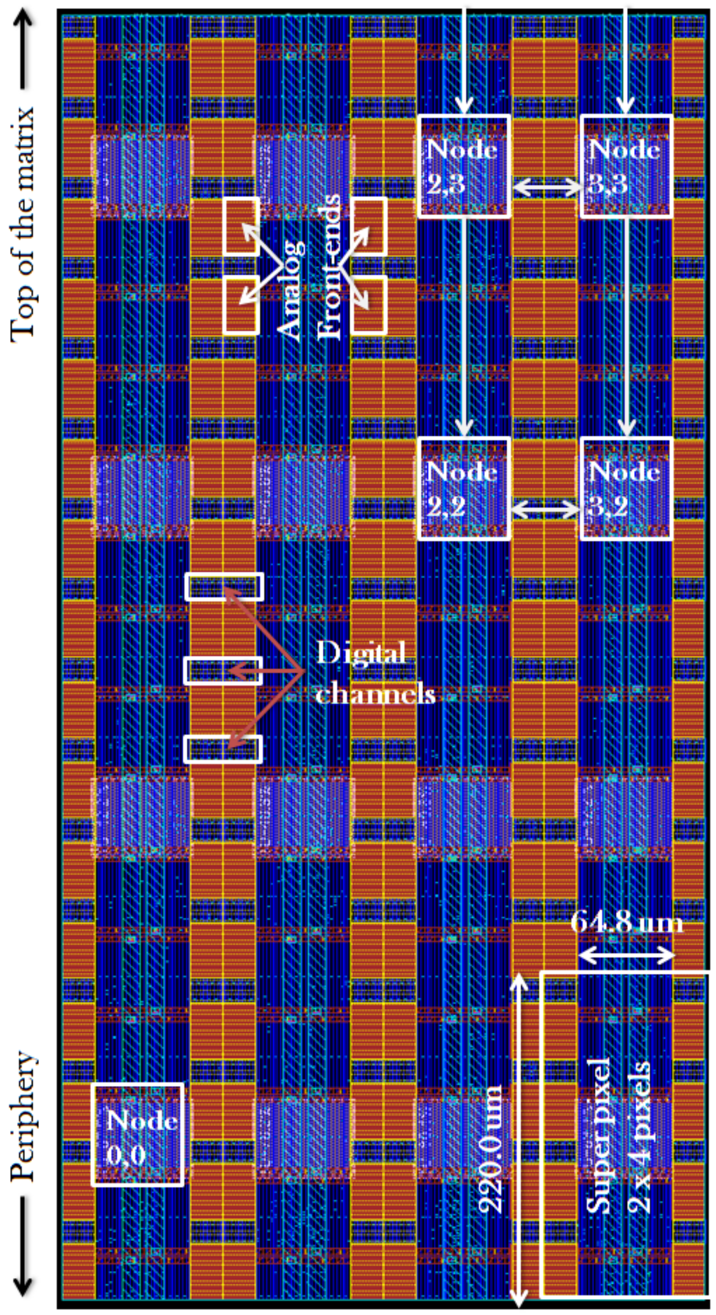


Figure 4.8: A layout of a network of 4 × 4 data nodes.

horizontally. As this would increase digital coupling to analog electronics, a test chip could be made to assess the impact of this coupling to the overall performance of a readout chip.

4.9 Super pixel dimensions

It was mentioned in Chapter 2 that multiple pixels can form a super structure called super pixel. When considering physical implementation details, there are several trade-offs in the choice of super pixel dimensions and the design methodology. In this thesis, automated PnR tools were chosen for the super pixel as well as for the digital front-ends. Unlike in [43, 25, 87] where a single pixel layout was used to build the pixel matrix, a larger block of pixels, a 2×4 super pixel, was used in this thesis. This means that a larger common area can be used for optimization, routing and placement of the logic. On the other hand, the pixels are not identical which can introduce systematic performance variations, especially in the analog front-ends.

As presented in Chapter 1, the analog front-ends are connected to the sensor by solder bumps on pads which utilize the topmost metal layer inside the ASIC. The load capacitance at the input of the analog front-end should be minimised because it increases the noise [98]. This capacitance should also be equalized across all front-ends to reduce channel-to-channel spread.

Three different super pixel floorplans are shown in Figure 4.9. The first one, a 2×4 structure was used for both the applications presented later in this thesis. This layout facilitates sharing of digital logic between 8 pixels. The clear advantage of any $2 \times N$ super pixel is that the clock distribution can be shared between two pixel columns. The analog front-ends can also share biasing lines and the power distribution network because the analog front-ends are back-to-back between double columns. The routing from the bump pads to the analog front-ends can be identical for each pixel. The main drawback of this layout is the placement of the pads on top of the digital logic. This can cause digital-to-analog coupling unless the pads are shielded using metal layers between the pad and the digital signals.

Figure 4.9b) shows a super pixel of 4×4 pixels. The advantage over Figure 4.9a) is the increased area for shared logic and routing. The clock distribution can also be shared between 4 pixel columns. This layout also has the disadvantage of the pads placed on top of the digital logic. However, the main drawback compared to the 2×4 layout is that the input capacitance for the analog front-ends is not the same which systematically increases the channel-to-channel spread. This capacitance can be equalized among the front-ends at the cost of increased noise and additional routing.

A third possibility is shown in Figure 4.9c) where analog islands are used to make the routing from the pads identical. In this case, the drawback is

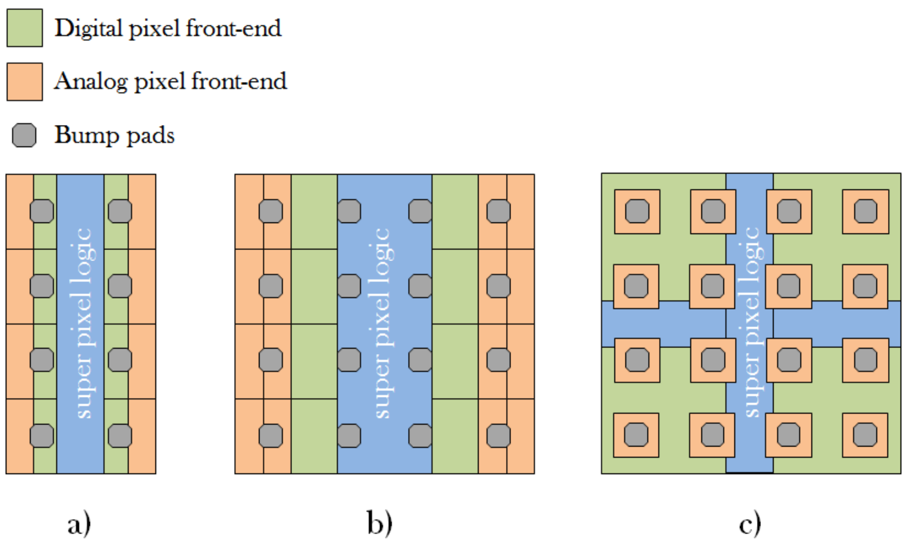


Figure 4.9: Super pixel floorplans: a) 2×4, b) 4×4 and c) 4×4 with analog islands.

that analog buses for biasing and power distribution must be routed over digital regions. Unless quiet logic is placed in these positions, this can cause coupling of digital activity to the analog signals thus increasing the noise. Some advantages for this layout are that it is possible to have digital communication between adjacent columns thus making it suitable for networks discussed in Chapter 2 and earlier in this chapter. The bump pads are also routed identically for each analog island thus minimising non-uniformities in the input capacitance to the front-end.

4.10 Concluding remarks

In resource-critical designs such as pixel readout chips, taking into account physical design aspects is crucial. Due to the effort required in physical design, architectures were first estimated using high-level techniques, and based on the performance in these simulations, some were chosen to be implemented and compared in terms of power, area and timing performance.

In this chapter the physical considerations of the readout architectures presented in earlier chapters were considered. It was seen that there are good alternatives to traditional bus-based data transfer architectures such as data fabrics and even networks in the pixel matrix. The data fabrics are especially attractive because they effectively localize the data communication between two adjacent blocks instead of having a global bus between several blocks. They also simplify the arbitration to a decision between two options instead of the group of all modules that utilize the bus or the fabric.

Chapter 5

Timepix3 ASIC

5.1 Motivation and requirements

In this chapter, a pixel readout ASIC called Timepix3 is presented. Before going into the details of this chip, a short motivation for designing the Timepix3 is given. The Timepix3 chip is a successor to the Timepix [25] readout ASIC. Timepix is a 65k channel HPD ASIC with charge/time measurement capabilities, and it can also operate in photon counting mode in which the number of detected particles withing a time window is recorded. It has been successfully used in several applications (for example [59, 99, 100]). It incorporates a global shutter-based operation and a full frame readout with a pixel-pitch of 55 μm. However, the chip also has some shortcomings:

1. minimum readout related dead time of 300 μs even for very low pixel occupancies (1 - 5 %)
2. lack of simultaneous time (ToA) and charge (ToT) measurement per pixel
3. no on-chip zero suppression (causing longer dead-time/lower hit rate)
4. resolution of the time measurement limited to 10 ns
5. no simultaneous event and charge measurement
6. no detections of overlapping hits in ToA and ToT modes

Most of these shortcomings were addressed using the architectural concepts presented in Chapters 2, 3 and 4. The implementation of Timepix3 therefore allowed verification of these concepts in hardware. Timepix3 reduces the dead time per pixel to 475 ns and provides simultaneous ToA and ToT measurements in every pixel (points 1 and 2). Dead time reduction is

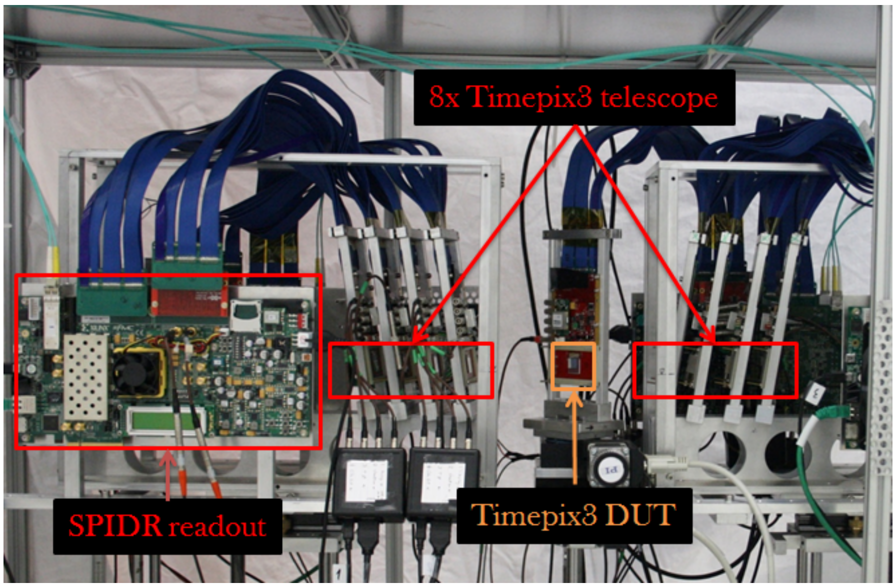


Figure 5.1: An application of Timepix3 in a particle tracking telescope. The photo taken from the LHCb upgrade testbeam program.

achieved using on-chip zero suppression (point 3) and a super pixel structure utilizing an intermediate FIFO buffer for hit storage. The targeted timing resolution is 1.5625 ns (point 4) and the measurement is made using a voltage-controlled oscillator (VCO) oscillating at nominal frequency of 640 MHz in each super pixel [101]. Due to its dynamic current consumption of 300 μA, the VCO is only switched on if a pixel in a super pixel asserts an enable-signal. A measurement mode where the ToT is integrated while the total number of hits is also calculated is implemented (point 5). Timepix3 also has hit overlap detection (point 6) in the digital front-end which can be enabled by sacrificing 4 bits of 640 MHz time measurement. Due to a design error in the pixel clock gating logic, this overlap detection is not working correctly, but it has been demonstrated that area-wise the feature could fit there.

One usage for Timepix3 is shown in Figure 5.1 as a particle tracking device. The telescope includes 8 Timepix3 ASICs on both sides, and one Timepix3 chip as a device-under-test. The telescope has been constructed as a joint effort between CERN and Nikhef.

The Timepix3 requirements are shown in Table 5.1. The chip is sensor-compatible with other Medipix family chips [43, 25, 42] by using a $55\ \mu m \times 55\ \mu m$ pixel size, and a pixel matrix of identical dimensions. ToT range is 10 bits at 40 MHz and the time stamp range is 14 bits for the coarse ToA and 4 bits for fine ToA. These measurements can be done independently of each

Table 5.1: Timepix3 requirements.

Pixel size	$55\ \mu m \times\ 55\ \mu m$
Number of channels	256×256
Operating frequency	40 MHz
ToT range	10 bits
ToA range	14 bits + 4 bits
Minimum time resolution	1.5625 ns (640 MHz)
Power consumption	< 1 W/cm^2
Hit rate per pixel	1.2 kHz (average)
Maximum hit rate	40 Mhits/cm^2/s
Output bandwidth	5.12 Gbps (SLVS)

other. In fact, in addition to measuring time, the 4 bits of fine ToA can also be used to improve the ToT resolution. A maximum power consumption of 1 W/cm^2 indicates that roughly 1 W/chip is reserved for analog, and 1 W/chip for digital logic. The average sustainable hit rate per pixel is 1.2 kHz which equals 40 Mhits/s/cm^2. This maximum output rate is limited by the periphery (and not the matrix) due to the maximum output bandwidth of 5.12 Gbps. There are 8 scalable low-voltage signaling (SLVS) output links in Timepix3 running at a maximum of 640 Mbps per link, producing 8b-10b encoded, serialized data streams. Each link has an independent packet data stream so the number of links can be adjusted to the requirements of the system (for example 1 gigabit ethernet (GBE) throughput, available tracks on a printed circuit board (PCB), maximum power consumption due to limited cooling capacity).

5.2 Architecture overview

The readout architecture of Timepix3 is a zero-suppressed (packet-based), trigger-less, continuous readout according to Sections 2.10, 2.7 and 2.9. The chip can operate with a duty cycle of 100 % (after configuration), meaning no separate readout cycle is needed, thus making it sensitive to hits at all times. This increase in duty cycle also reduces the occurrence of measurement errors which happen when a discriminator output is high at the time of the shutter opening, because the shutter can be kept open during the readout. The pixel matrix consists of 128 double columns, with 64 super pixels each. Each super pixel contains 2×4 pixel front-ends.

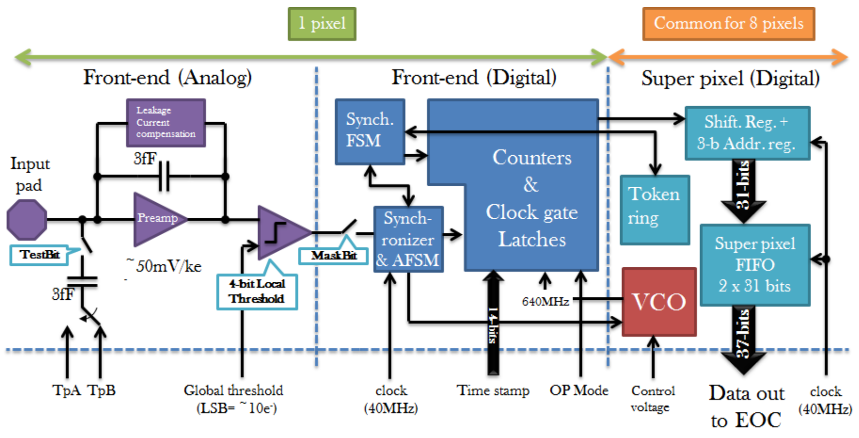


Figure 5.2: A schematic of the front-end of Timepix3.

5.3 Front-end description

The front-end of Timepix3 including the super pixel is shown in Figure 5.2. It consists of 8 analog and digital front-end pixel circuitry connected to a shared super pixel logic. The super pixel input and output rates were chosen according to the simulations presented in Chapter 2, but also taking the physical constraints into account. The super pixel is used only for readout purposes, not for reducing the data rate by grouping several hits into one packet. Because each pixel must store 28 bits of information, grouping multiple hits into one packet would make the length of the packet more difficult to handle than single-hit packets. Even if the time stamping information (14 bits) was shared between 8 pixels, each pixel would require 14 bits inside the grouped packet. A fixed size packet would have 126 bits of information excluding the address bits. Setting maximum number of hits per packet to 3 (see Section 2.10 would decrease the total length but would require additional packet formatting circuitry within the super pixel.

5.3.1 Analog front-end

The hit processing starts from the analog front-end, which receives and amplifies the current pulses arriving from the sensor before comparing the output value of the amplifier against a pre-set discriminator threshold. If this value is exceeded, the discriminator output generates a logic one which is sent to the digital front-end for synchronization. The analog front-end consists of a pre-amplifier with leakage current compensation, a 3 fF feedback capacitor and a discriminator. It also contains circuitry for test pulse injection and local threshold tuning of 4 bits. These bits can be programmed per pixel basis. The analog front-end of the Timepix3 has been described in

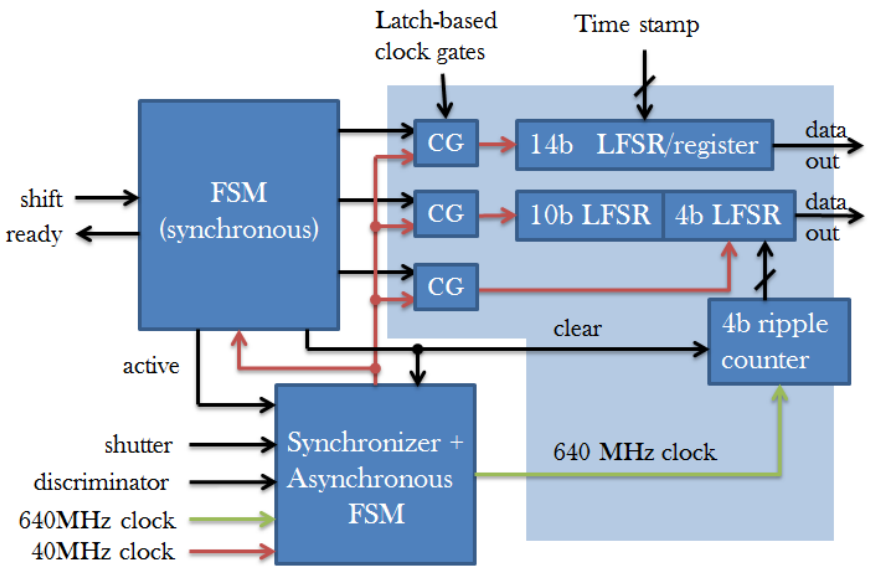


Figure 5.3: A block diagram of a digital pixel of Timepix3.

detail and characterized in [16], and is not discussed here.

5.3.2 Digital front-end

The digital pixel logic of Timepix3 is shown in Figure 5.3. This pixel is a building block for a larger structure of 2 × 16 pixels which is repeated to form the full matrix.

The AFSM is implemented to reduce dynamic power consumption of the pixel. Because one flip-flop used in the standard cell library consumes approximately 1 μW running at 40 MHz even when its output value does not change (internal power), a two flip-flop synchronizer is not used. Using an AFSM, dynamic power consumption is reduced to < 0.5 μW per pixel when there is no hit activity present. This technique is deployed similarly in digital front-ends in many other HPD readout chips [25, 35, 75]. The AFSM is essentially a clock gate, having two inputs, enable and clock and outputting a gated clock signal. The main difference to a latch-based clock gate is that it does not produce any glitches on the gated clock output regardless of the arrival time of the input clock and the clock enable-signal. This behaviour is frequency-dependent and must be verified in analog simulations. No glitches have been observed when simulated at the targeted operation frequency of 40 MHz.

The AFSM provides the gated clock to the synchronous FSM which controls the counters and the readout of the pixel. The state chart of this

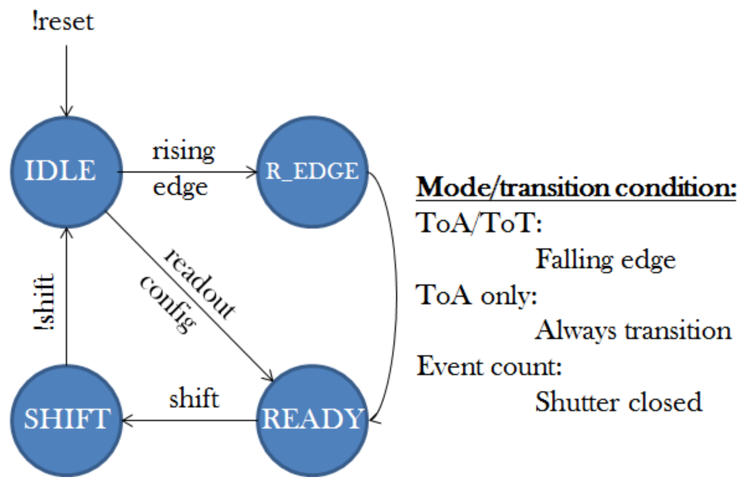


Figure 5.4: Synchronous state machine of the digital front-end of Timepix3.

FSM is shown in Figure 5.4. In any of the three modes, the FSM is not clocked and is idle until the first rising edge of the discriminator. The FSM is activated when the rising edge of the discriminator output arrives, and it transitions into the state `R_EDGE`. See Figure 5.5. This state is held until one of the conditions described in Figure 5.4 is met, depending on the operation mode of the pixel. For example, in ToA/ToT mode the transition happens when the falling edge of the discriminator arrives. The processing is different in all modes but the readout operation is performed similarly. It can be seen that the pixel cannot transition back to the state `R_EDGE` once it is ready. This means that only one shutter-opening can be recorded in the event-counting mode, and in the other two modes arriving hits are discarded until the current data has been read out. A pixel can accept a new hit every three clock cycles, thus having a dead time of 75 ns, but this is possible only in the event-counting mode due to the additional dead time contributed by the super pixel readout logic. In the event-counting mode, the pixel need not be read after every hit. After the readout operation, the FSM returns to the `IDLE` state and the AFSM disables the clock again. During a configuration data readout, the FSM transitions directly from `IDLE` and `READY`, and the configuration data can be read out after this similarly as the event data.

A standard technique of latch-based clock gating is used to provide gated, glitch free clocks to the pixel counters. This ensures that the counters consume power only when a pixel is processing a hit. Another well-known advantage is that the feedback muxes are removed from the flip-flops thus resulting in reduced area. The reset functionality in the counters is implemented by shifting the reset values into counters. This reduces the area required for the counters but increases the duration of reset. The counters

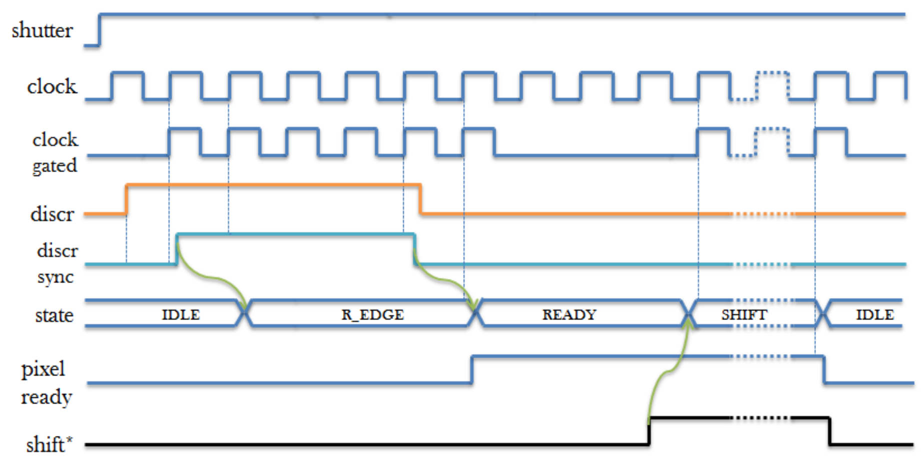


Figure 5.5: Timing diagram of the digital front-end of Timepix3.

are implemented as LFSRs instead of binary counters because this technique typically saves at least 30 % of the area [89]. LFSRs also run faster than synchronous binary counters but the speed is not a critical attribute in this case. Binary ripple counters could also be used if the bits in the counters were read in parallel, but configuring a ripple counter into a shift register for serial readout requires an extra multiplexer per flip-flop thus increasing the area footprint.

To further optimise the design, the design hierarchy was flattened inside the synthesis tool, and an optimal state encoding in terms of area for the synchronous FSM was iterated. This was done by synthesizing the pixel using all possible combinations of state vectors. Another optimization was to assign macros to all values of control signals, then try different combinations of these values to find which needs the smallest area. Using the macros instead of hard-codes values, the active level of these signals could be rapidly changed. An example of this is shown in the listing below:

```
`define PIXEL_SHIFT_ENABLE_ON   1'b0
`define PIXEL_SHIFT_ENABLE_OFF  1'b1
```

Before the optimizations, the estimated cell area was 1575 μm^2 and the estimated area for routing 2184 μm^2. After the optimizations, the cell area was 1536 μm^2 (a reduction of 2.5 %) and the area for routing 2089 μm^2 (a reduction of 4.5 %). Although the impact of these optimizations seems to be small, in fact the design could not be placed and routed without them. Before starting the routing, the placement density was already over 90 %. For example, it was not possible to add overflow control to the 14-bit counter/register anymore. The overall transistor density (900 ktransistors/mm^2) was over two times larger than in [75], for example. The length of the pixel

RTL description is approximately 1300 lines of Verilog hardware description language (HDL).

5.4 Super pixel

The super pixel logic of Timepix3 is shown in Figure 5.6. This logic operates as a data concentrator and readout logic for 8 digital front-ends. The readout is controlled by an FSM with a counter, and it reads one pixel at time. The arbitration is done using a token ring, in which the token can travel between any two locations in one clock cycle.

Because the super pixel logic utilizes standard cells that are optimized for area, the number of routing channels in the cells is also decreased. To compensate for this loss of routing resources, bits from pixels are transferred to shift registers using a 2-bit bus only. This adds 14 clock cycles to the dead time of the digital front-end. The dead time is 19 clock cycles in total, consisting of the shift time (14) and time to initiate the shift (2) and reset the pixel (3) after reading the data. The parallel readout of all 28 bits from a pixel was not possible due to the limited routing resources available.

A linear FIFO is used to store up to two pixel data packets. This FIFO architecture is smaller in terms of area than a FIFO with multiplexers inferred at the inputs and outputs. Even if the FIFO is full, the super pixel can still initiate a shift-operation from a pixel and store this data into the shift registers until a slot from the FIFO is freed for writing. The FIFO is read by a TX FSM which requests the bus access from an EoC block. After the request, the FSM must wait until it receives the token from the previous super pixel block (or from the EoC if it is the first in the ring). Each packet is split into 4 words of 10 bits to reduce the number of wires required for the bus. A two-phase handshake is performed for each word between the TX FSM and an RX at the EoC.

Simple state charts for two FSMs required in the super pixel logic are shown in Figure 5.7. Some conditions and transitions are omitted for clarity. As described, the Shift FSM controls the readout operations from pixels, and starts shifting data from a pixel selected by the token ring after it receives a request. After the counter reaches value 13 ($shift\_done$), the pixel data is written into the super pixel FIFO unless the FIFO is full. In that case, the FSM waits until the FIFO has a free slot to write into. This FIFO acts as a buffer between the Shift FSM and the TX FSM. The TX FSM starts its operation when the $fifo\_empty$ -flag is deasserted. After this it requests the token and starts the bus transaction after having received the token. The transaction is complete when the $send\_done$ -condition is reached.

The readout architecture of one double column of Timepix3 is shown in Figure 5.8. The full column consists of 64 super pixels connected to an OR-

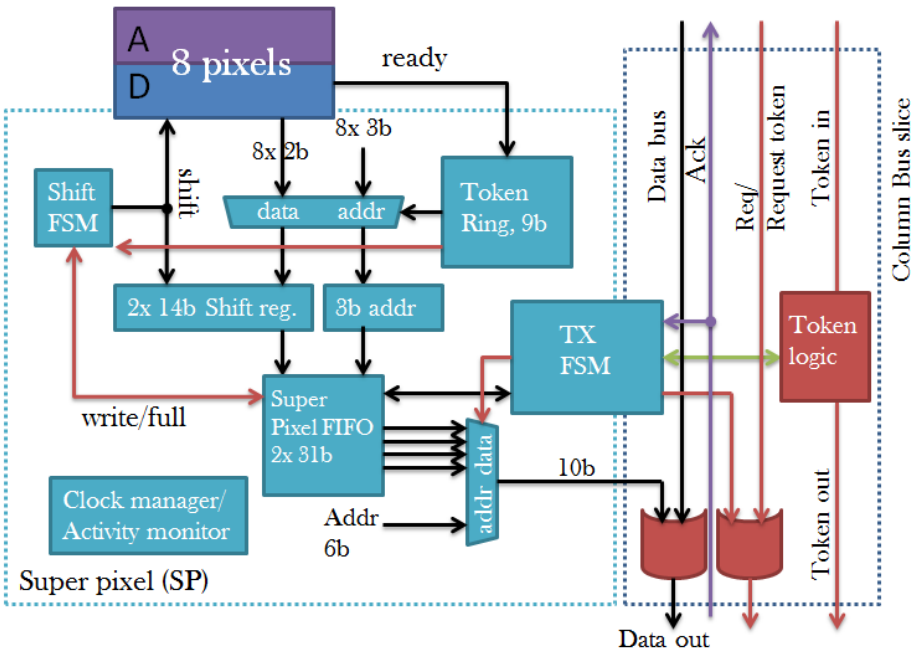


Figure 5.6: A block diagram of a super pixel of Timepix3.

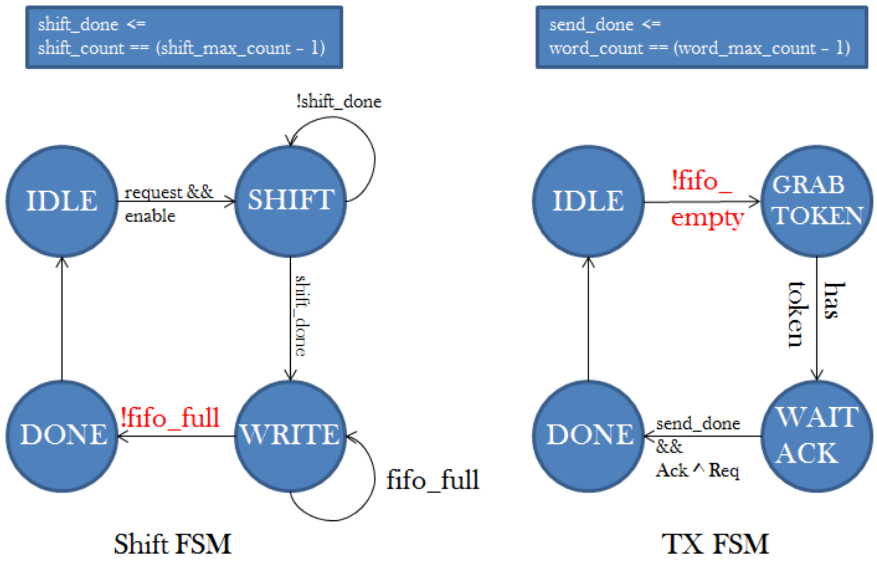


Figure 5.7: State diagrams for the super pixel FSMs.

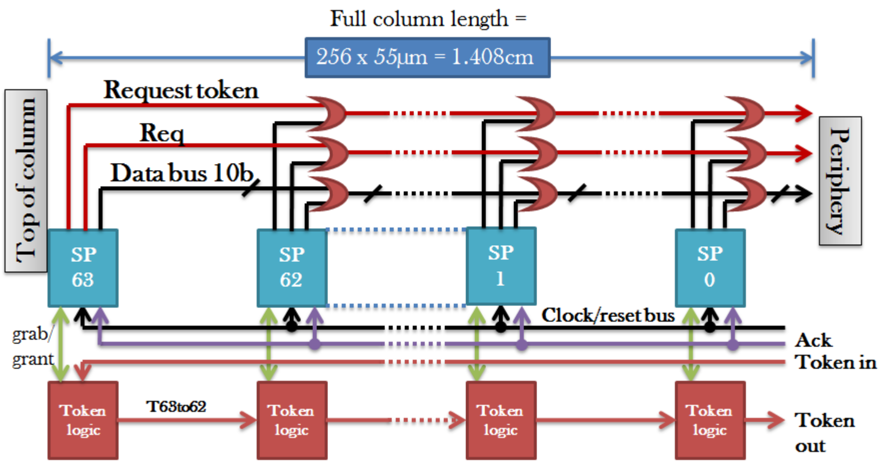


Figure 5.8: A block diagram of the double column of Timepix3.

based bus using a 2-phase asynchronous handshake protocol [102]. Data are single-rail encoded meaning only one wire per data bit is used. This imposes strict timing relationship between request- and data-signals. The data must always be ready before the request arrives to the receiver. This protocol works in a globally-asynchronous locally-synchronous (GALS)-manner [103] because each super pixel uses a locally synchronous clock but the communication between the super pixels and the receivers is asynchronous. The timing requirements for the request and data signals have been defined by using a virtual clock in the SDC file.

Figure 5.9 shows the timing diagram related to a bus transaction. The transaction is started when a requesting TX FSM receives the token (*token_in*). Because the handshake is performed using transitions on *Ack* and *Req* , an XOR-operation can be used to detect the transitions. Both the TX and the receiver use a double flip-flop synchronizer to avoid synchronization failures. The clocks on both ends can be completely independent, and can come from non-related sources. Note that the receiver and the token are also independent of each other. The token always operates in the same clock domain as the TX. However, *Request* must be synchronized before giving it to the token logic. This is required because *Request* is an OR-function of 64 bits and the timing for this signal is not guaranteed.

5.4.1 Choice of super pixel dimensions

A geometry of 2×4 pixels per super pixel has been chosen for several reasons. Firstly, the intermediary FIFO used to reduce the dead time could not be used per pixel-basis due to a limited area available. However, even by sharing

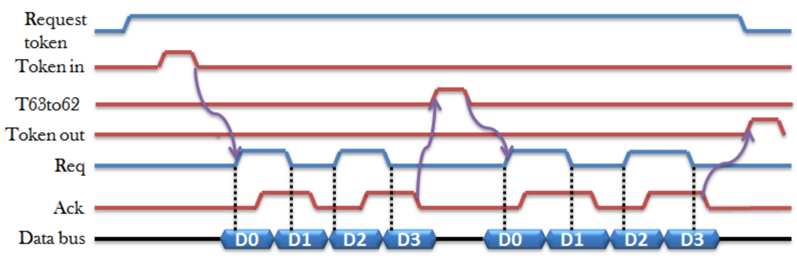


Figure 5.9: Timing diagram of the column bus of Timepix3.

this buffer between multiple data producers, as was shown in Chapter 2, the efficiency can be improved. The column bus timing is also improved by connecting only 64 super pixels to the bus instead of having 256 or 512 pixels connected to the same bus. Because the arbitration of this bus is based on a synchronous token ring, the traversal time of the token through the ring is also reduced.

Secondly, a larger super pixel was avoided to limit the routing distances between digital blocks. Because the standard cells used are optimized for area by decreasing the width of the transistors in the cells, they are slow and have lower driving capability compared to commercial libraries in the same technology. By limiting the super pixel to $110\mu m \times 220\mu m$, it is ensured that no signal from pixels needs to be driven further than this distance. By restricting the size to two pixels in the row direction, the input capacitance to the analog front-end is minimised and uniform across the pixels. Unequal capacitances increase the systematic offsets between analog front-ends and increasing the input capacitance adds additional noise to the front-end [98].

As was discussed in Chapter 4, there are a number of trade-offs when choosing the super pixel size and the design methodology. For Timepix3, automated PnR tools were used for the super pixel as well as for the digital front-ends. More precisely, the matrix was constructed from a block of 4 super pixels containing a rectangle of 2×16 pixels. This has the flexibility of faster modifications to the layout and direct correspondence of the layout with the RTL description. The drawback is increased mismatch in the analog front-ends because the layout of the digital pixel front-end is not identical for all pixels anymore. Another drawback, which was already observed in simulations, are timing mismatches in the digital logic while measuring the fine time stamp. To alleviate this problem, a synchronizer and clock gating block for the digital front-ends was created as a macro block, and placed at the same position in all pixels. Despite this optimization effort, the binning of fine time stamps measured from the manufactured Timepix3 was not equal in all pixels in 2×16 pixel structure.

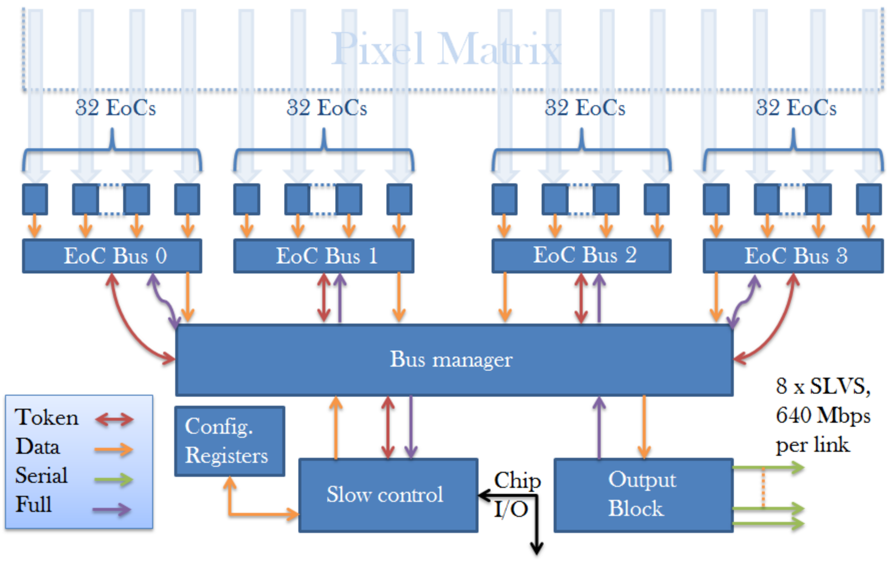


Figure 5.10: The digital periphery of Timepix3.

5.5 End-of-Column and Periphery

The digital periphery of Timepix3 is shown in Figure 5.10. The data path is divided into 4 buses and 4 token rings. 32 EoC blocks are connected to each bus-ring combination. The connections between EoCs and each bus segment (0 - 3) are omitted for clarity. The main blocks of the periphery are EoC block, the bus manager, the output block, slow control and command decoder and the configuration registers. There are other control units which are not shown in Figure 5.10, such as power pulsing control and a command decoder for pixel matrix commands.

The EoC block has two main functions. The first is to receive data packets from columns by synchronizing the data to the periphery clock domain after a completed handshake. An EoC block adds a double column address to each packet for later identification. Packets are stored in a FIFO until the EoC is granted access to a 48-bit periphery bus. After gaining access to the bus, one packet from the FIFO is sent to the output block for serialization.

The second function is to perform control operations for the pixel matrix. Reading and writing configuration data to the matrix is handled by the EoC command decoder, resetting the pixel matrix is sequenced by EoC blocks, and readout modes are enabled and disabled through this logic. As a last control function, the test pulse injection on a column basis is controlled by a configuration register in the EoC block. The clock gating to the column is also controlled by the EoC clock manager, which shuts down the clock propagation during sequential readout operation from columns which are not being read out. The main purpose of this logic is to reduce the power consumption by not reading out all columns simultaneously. This does, however, sacrifice readout speed, but is useful if Timepix3 is connected to a system that cannot handle the full output bandwidth of 5.12 Gbps. The number of columns read out in parallel can be chosen with a programmable mask.

All data packets from EoC blocks pass through a bus manager controlling the 4 EoC buses. These buses consist of 4 rings each of 32 EoCs and a central arbiter which chooses the next ring. The arbitration within each of the rings is done using token arbitration. This implies that the central arbiter has no control over the token while it is inside one of the 4 rings. The token can travel from one station (a flip-flop) to the other within one clock cycle (80 MHz) in a ring, but it takes one extra clock cycle to change the ring. This is done to improve the timing of the bus, and to be able to run it at 80 MHz, effectively delivering 3.84 Gbps. The OR-based bus follows the same structure as the token rings, meaning that 32 EoCs are connected to the same bus. All control packets from the slow control logic also propagate through the arbiter meaning that the control data is always mixed with the data packets.

The bus manager passes received packets on to the output block. This block can control the data flow by setting $Full$ -flag, if it cannot keep up with the stream of incoming packets. This can happen if the clock frequency of the links is lowered from 320 MHz double-date rate (DDR) or if some links are disabled. The output block also encodes the data using 8b/10b encoding to create a double column (DC)-balanced data stream from which a receiver can recover the clock. Similarly, as in [56], the number of enabled serialisers in the output block can be configured to accommodate applications with higher or slower hit rates. This also means that the Timepix3 can be operated with one output link only, if the readout system cannot incorporate more links due to area restrictions on the chip board.

Timepix3 also has an on-chip PLL generating a 320 MHz clock from an input clock of 40 MHz. Besides generating the 320 MHz needed for serialization of the pixel packets, it generates a control voltage for VCOs that are used for fine time stamps measurements. This PLL has been described in detail and has been characterised in [104].

5.6 Physical implementation

For the physical implementation of Timepix3, a 130 nm commercial CMOS process was used. The CMOS technology and its 8-layer metal stack were already described in Chapter 4. In the design of digital logic for the pixel matrix of Timepix3, layers up to M4 were used in the routing of digital signals and the upper layers were completely reserved for the power distribution. Two different standard cell libraries were used, with row height 2.4 and 4.8 μm respectively. The former option allowed much higher integration of gates into the pixel matrix by reducing the width of the transistors used in the cells with a corresponding reduction in switching speed. These cells were also created using high threshold voltage transistors to reduce the leakage current of the matrix. The larger cells were used at the periphery of the chip where higher frequencies were required (up to 320 MHz). Because the cell count at the periphery was an order of magnitude smaller than in the pixel matrix, no leakage power optimization was done.

Timepix3 was implemented using a mixture of automated digital design tools and analog custom layout design flow. The analog and digital developments were done in parallel. The following steps were carried out to build the layout of the full chip:

1. Analog front-end layout was done in a full-custom manner.
2. The VCO used in the super pixel was done in a full-custom manner.
3. A synchronizer block for the digital front-end was created using PnR tools.
4. The layout of 4 super pixels (2 $\times$ 16 pixels) was created using PnR tools, 4 VCO layouts and 32 synchronizer layouts. The global signals such as the clock and the time stamp bus were fixed to specific positions.
5. The layout for a full column was created using 64 super pixel layouts and 512 analog front-end layouts. The columns on the both edges of the matrix were done separately.
6. The matrix was created by replicating the column layouts 128 times.
7. The EoC block was created using PnR tools.
8. The layout was created for a 640 Mbps channel in the output block.
9. Ready-made intellectual property (IP) blocks, such as DACs and a bandgap reference, were used for analog peripheral components.
10. A PLL was created in a full-custom manner.
11. The periphery was created using PnR tools, instantiating all layouts and the IO pads.
12. The matrix and periphery layouts were combined in an analog layout editor. This is also called analog-on-top approach.

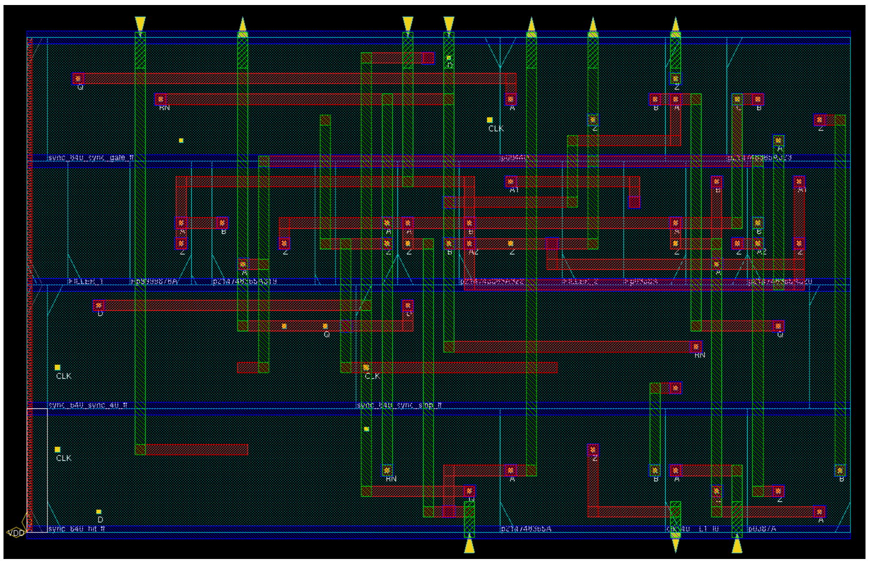


Figure 5.11: The layout of the front-end synchronizer in Timepix3.

In this section, only points 3, 4 and 7 are discussed. However, point 11 was also relevant for this thesis because architectural optimizations of the RTL code were required to meet the timing constraints on the periphery token and bus. As was shown in Figure 5.10, 32 EoC blocks were connected to one bus segment. This number was mainly dictated by the physical design constraints and the clock frequency of the bus (80 MHz). Because no tristate gates were used in the design, an OR-based bus was used to merge data signals from 32 blocks into one bus. As was shown in Chapter 4, for a 64-module OR-based bus, the worst case delay was 12.3 ns using the most power efficiently sized transistors. To guarantee the timing with a clock period of 12.5 ns, a 32-module bus was used.

In Figure 5.11, the layout of the digital pixel front-end synchronizer block after PnR is shown. This block has been created separately to reduce the timing mismatches between pixels instead of placing and routing it with the super pixel layout. The dimensions of the block are 9.6 μm $\times$ 16.0 μm, and the local routing has been restricted to the three lowest metal layers. This is done to facilitate easier integration of this block at the next level of the hierarchy where higher level metal layers are required in global routing. The power is distributed using M1 horizontal stripes (shown in dark blue in Figure 5.11), and are connected to the column-level power grid when integrated with the layout at the next level of hierarchy.

The layout of one super pixel is shown in Figure 5.12, and it has been rotated 90 degrees to the right. The area taken by this block is 110 μm $\times$ 220 μm. All global routing is shown with the horizontal metal lines (M4). The

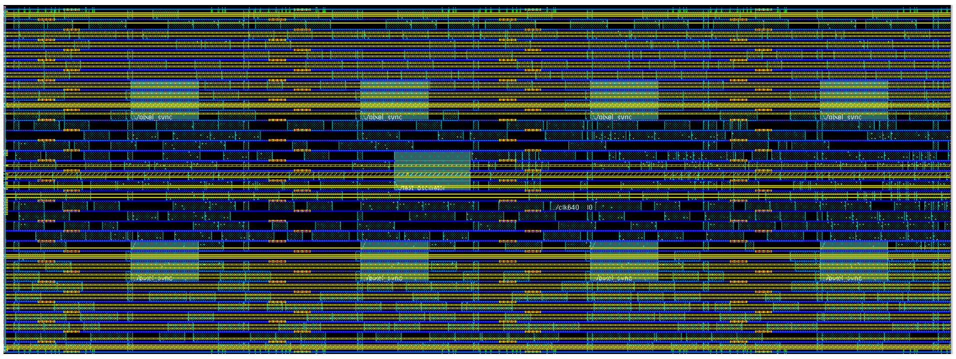


Figure 5.12: The layout of a super pixel of Timepix3.

most routing area is taken by the two 14-bit time stamp buses, and the clock signal is routed using a wider wire (0.4 μm) to reduce the resistance of the wire. To ensure identical signal distribution, the routing of the global signals was carefully designed and their placement done by hand rather than with the automatic router. The buffering of all signals, except the clock signal, is done at the EoC block, and thus is not shown. Note that all signals going to the EoC (from right to left) are not manually routed.

There are 8 synchronizer blocks and one VCO within the super pixel layout, which also contains all counter logic for pixels and the readout logic of the super pixel. Due to the large aspect ratio, the full building block of the pixel matrix, a layout of 4 super pixels, cannot be shown. The super pixel layout is not identical for all 4 super pixels due to automatic PnR. Some features such as macro blocks, global routing and the power distribution are implemented manually, and are identical in all super pixels. The power distribution which is not fully shown (see Figure 5.13), is done using M5 in a row-direction (up-down) and the two top metals, M7 and M8 in a column-direction (left-right). Standard cells were placed freely by the digital placement tool to optimise the area, and the gaps between standard cells were mostly filled with decoupling capacitor cells. The estimated distributed decoupling capacitance from these capacitors is 9 nF. Similarly, in [74] only 90 % of the area was used by the analog and digital functionality, so decoupling capacitors were used to increase robustness against varying current consumption.

The power distribution of the column is shown in Figure 5.13. The top metals M7 and M8 are 25 μm wide to minimize resistive voltage drops on the power supply lines. The M5 routing is 4 μm wide and is connected as a grid over the analog front-ends. The width of the top metals is limited by the bump pads which are located between two M5 pairs, and almost on top of the synchronizer blocks. Despite all the efforts made to shield the pads from digital activity, increased noise was observed when the time stamp bus

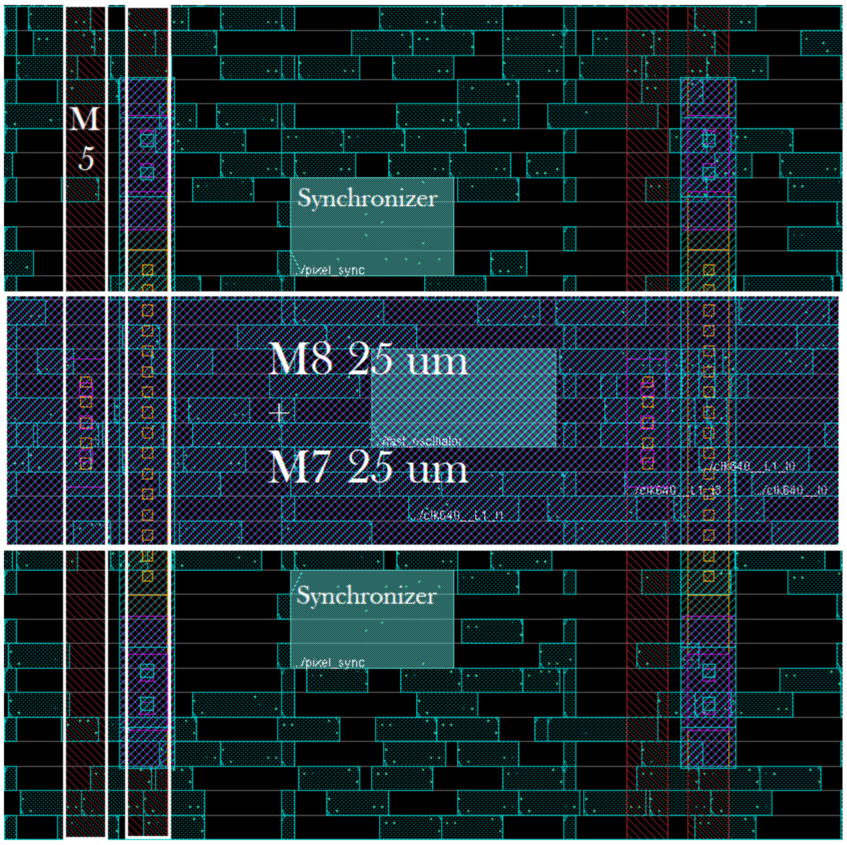


Figure 5.13: A segment of the digital power distribution for a column.

was turned on. This increase is quantified and analyzed in the following sections in this chapter.

5.7 Test setup

The measurements from a manufactured chip were performed to assess the performance of the architecture. An FPGA-based readout system called Speedy PIxel Detector Readout (SPIDR) [105] was used for all the measurements. The readout board features a 10 Gb Ethernet (10GbE) for reading out the chip at full of speed of 5.12 Gbps. This test setup for measurements of Timepix3 is shown in Figure 5.14. All measurements were performed from a single chip only. The Timepix3 was mounted to a circuit board and the chip I/O wire-bonded to the board.

All tests for measurements were written using Python scripts and the test application programming interface (API) was built on top of a library of C++ functions. The development of C++ library and the test API was

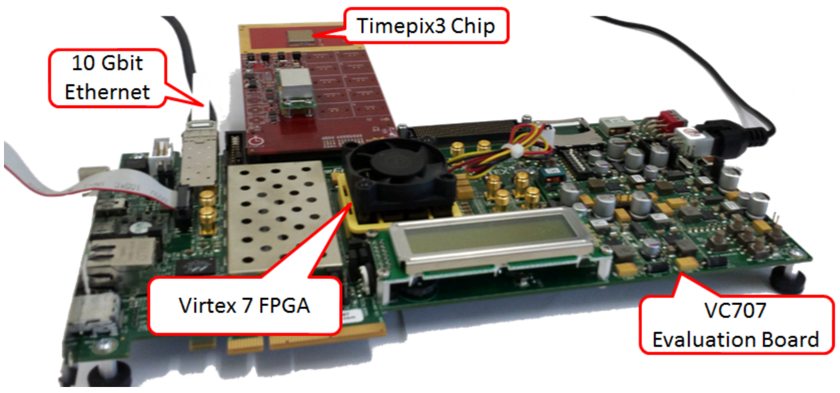


Figure 5.14: The test setup used for the measurements of Timepix3.

not part of this thesis. A soft-core Leon processor is used inside the FPGA to enable some functions to be implemented in software. To control the temperature of the chip during the measurements, a large external fan was used, and the temperature variation observed during the measurements was from 55 C° to 58 C°.

5.8 Power consumption

The digital current consumption was measured using a 4-wire measurement technique. The board-level power supply is bypassed, and the chip connected to an external power source with measurement capabilities. The current drawn from the source by the chip can be sampled via a standard commands for programmable instruments (SCPI)-bus.

Based on analog transistor-level simulations, the power consumption for the clock distribution of the pixel matrix was estimated to be 220 mW, at room temperature and 1.5 V. The transient current drawn by a block of 16 double columns is shown in Figure 5.15. The rising edges are denoted as $R0 - R14$ and the falling edges correspondingly as $F0 - F6$. Phase difference between two consecutive double columns is 22.5 degrees, or 1.5625 ns when the clock period is 25 ns. The peak current consumption occurs when a rising and a falling edge coincide, and is approximately 18 mA for 16 double columns. This equals to a maximum current of 144 mA for the full chip.

The power consumption of the clock tree was measured from the chip with different clock frequencies of 20, 40, 80 and 160 MHz. Firstly, the digital power consumption of the periphery was measured while the shutter was closed. In this way, clock distribution to the pixel matrix was gated and consumed no power. After this, the shutter was opened and the power

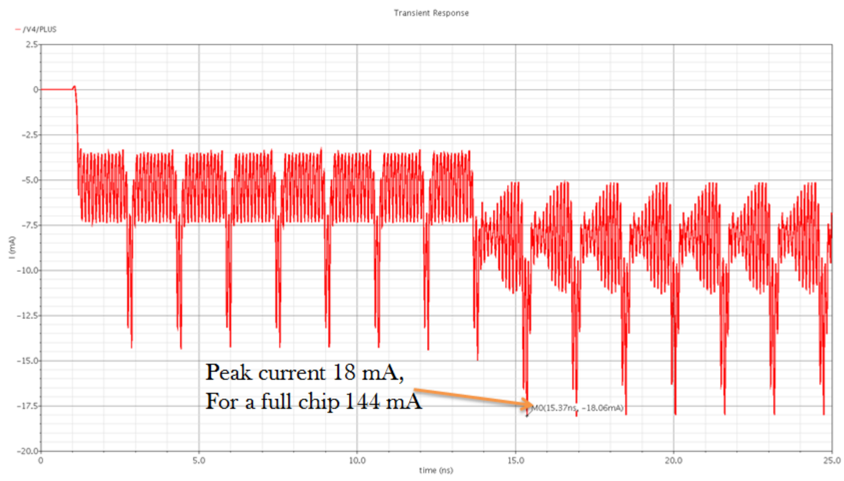


Figure 5.15: The dynamic current consumption for the clock distribution of 16 double columns.

consumption measured again. The analog front-ends were masked to ensure that no other activity than the clock was present in the matrix. By subtracting the two figures, a number for the clock distribution of the pixel matrix was obtained. The digital current consumption of the chip is shown in Figure 5.16. The measurement at 40 MHz (142 mA) is in agreement with the transistor-level simulation. The increase in the current consumption is linear with the clock frequency as expected. The power consumption of the ToA distribution in columns was estimated to be less than 55 mW (40 MHz, 1.5 V, 25 C°) based on analog schematic simulations. The measured current drawn by ToA counters can be calculated by subtracting the consumption of the periphery and the consumption when the ToA counters are enabled. At 40 MHz, this current was measured to be 44 mA, or 66 mW. This consumption is slightly higher because the simulation estimate does not include the actual counters at the periphery or the registers at the EoC.

5.9 Crosstalk and digital-to-analog coupling

When integrating the analog and digital parts in proximity of each other, digital-to-analog coupling is an issue. The typical ways that the noise can couple to analog sections is either via a noisy substrate or by direct metal-to-metal capacitive coupling. In the digital domain, the first problem can be addressed by reducing the voltage drops in digital power supply and ground lines by using a low-resistance power grid. Also, avoiding large current spikes reduces the substrate bounce. The second problem can be addressed by using some metal layers for shielding the nodes that are seen as vulnerable

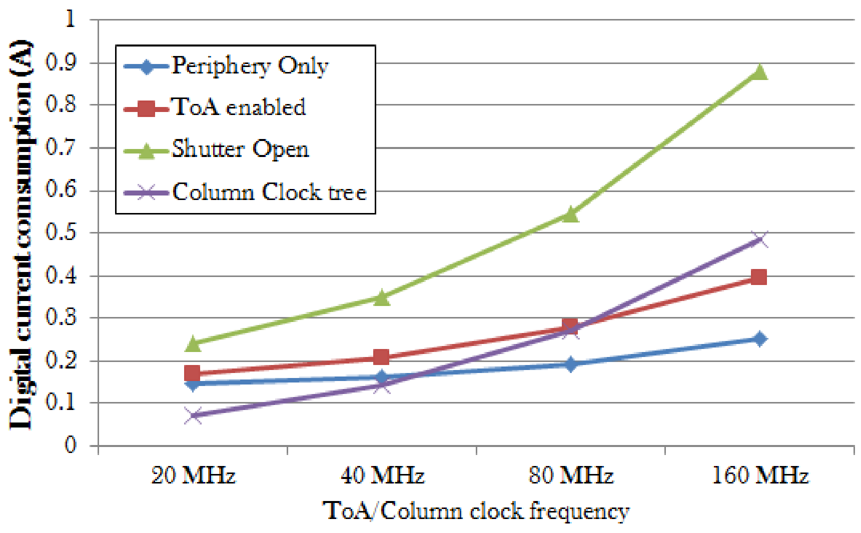


Figure 5.16: The current consumption measured from the Timepix3 chip.

to the coupling.

The difference in the minimum achievable threshold in two pixel operation modes and readout modes is shown in Figure 5.17. The x-axis indicates the threshold and the y-axis indicates the number of pixels having hits due to noise. A minimum threshold of 400 e^- was obtained when using event counting and integral ToT mode. In this mode there is no timing measurement, and the 14-bit time stamp buses are not toggling in the column. Readout of the data was also performed using a sequential readout mode meaning that the readout was disabled during the data acquisition. When the same measurement was repeated in simultaneous ToA/ToT mode using continuous data acquisition, a minimum threshold of 500 e^- was reached. By turning off the ToA counters at the periphery, it was verified that the increase in the minimum threshold was caused by the activity of the ToA counters, and not the usage of the continuous readout mode.

5.10 Column architecture characteristics

Figure 5.18 shows the packet readout rates from one double column in Timepix3. 256 pixel packets have been read out, and the periods between two successive packet arrivals are measured. The rate plotted in Figure 5.18 is then inverse of these periods. These packets contain configuration data and the packet traffic has been created using a slow control command. Due to the logical structure of the two token rings, one inside the super pixel and the other connecting super pixels together, the pixel 0 of the super pixel 63

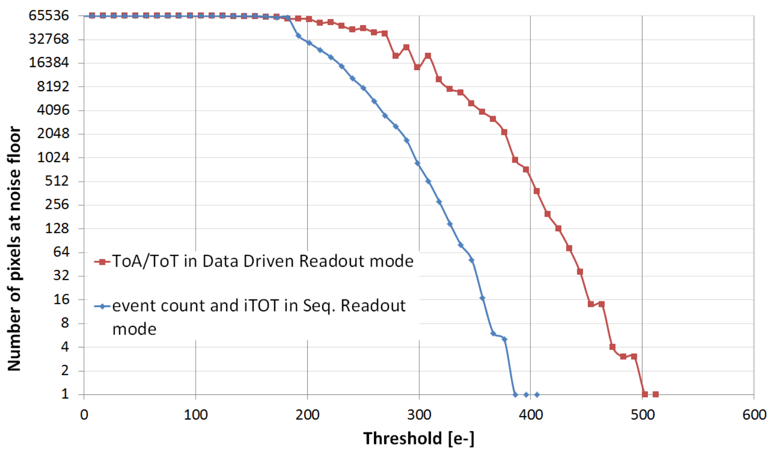


Figure 5.17: The impact of the ToA counter to the minimum threshold.

is read out first. This corresponds to packet 0 in Figure 5.18. The packet 1 then corresponds to the pixel 0 of the super pixel 62 and so on.

Because the data transmission starts from the top of the column, the rate is lower at first due to the larger wire- and gate-delays. The rate noticeably increases towards the packet number 63 (pixel 0, super pixel 0) and then drops again for the packet 64 (pixel 1, super pixel 63). The same periodic pattern is seen four times because there are 64 super pixels, and each super pixel sends only one packet for each of these four patterns. The plateaus in the plot are regions where the latency in clock cycles stays constant because all super pixels in the plateau are within one clock period in terms of timing of the handshake signals. In the transition region, the clock signal and the handshake signals coincide. Because there are 4 transitions per handshake and 4 transactions per packet, there are 16 possible events to synchronize. In the case of setup- or hold-time violation, the simulation model of the flip-flop randomly chooses the next value. The values of packet rate measured on the chip match exactly the simulations with the typical parameters.

Average simulated rates for a column using different super pixels (SPs) are also summarized in Table 5.2. The simulations have been performed using three process corners presented in Chapter 4: slow corner (SS), typical corner (TT) and fast corner (FF). The first two columns indicate that the deterioration in performance caused by wire- and gate delays is approximately 10 %. In the case of a single super pixel being hit, the largest delay is caused by the synchronous token ring which takes 64 clock cycles for a full round trip.

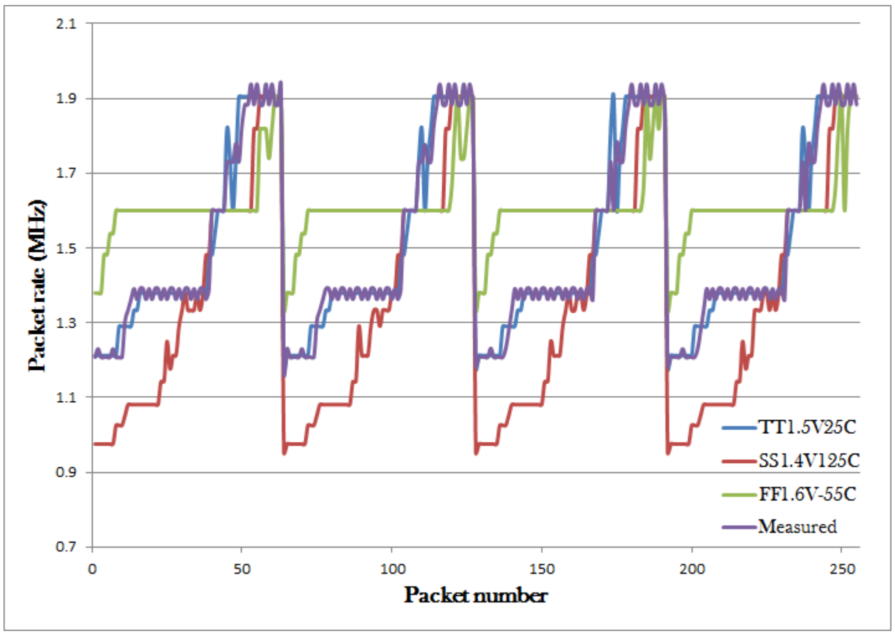


Figure 5.18: Measured/simulated readout rates of packets from a double column of Timepix3. The rate is an inverse of arrival times observed between two consecutive packets.

Table 5.2: Simulated average readout rates of super pixels 0 and 63, and average of one column in Timepix3.

	Corner		
	SS/1.4V/125C	TT/1.5V/25C	FF/1.6V/-55C
SP 0 only	450 kHz	450 kHz	450 kHz
SP 63 only	410 kHz	450 kHz	465 kHz
SPs 0-63	1.36 MHz	1.50 MHz	1.60 MHz

5.11 Main limitations of Timepix3

One of the main limitations of Timepix3 is that because each pixel has 28 bits of information, a bandwidth of 5.12 Gbps is only sufficient to deliver approximately 80 Mhits/s/chip, or 40 Mhits/s/cm^2. Two ways to improve the rate are increasing the output bandwidth and reducing the number of bits per pixel. All other things being equal, changing either of these parameters increases the total achievable hit rate linearly. For example, by increasing the output bandwidth to 20.48 Gbps and reducing the number of bits in the output packet from 48 bits to 40 bits, 512 Mhits/s/chip could be delivered. Knowing that the power efficiency in modern serialisers, in 65 nm CMOS technology for example, can be easily below 1 mW/Gbps using SLVS [106], the power budget of Timepix3 would allow many more additional serialisers. On the one hand, the limiting factor would then be the available I/O pins, but on the other hand, the power consumption of the internal readout architecture would also increase because it would have to accommodate much higher rates.

Another limitation is that the pixel hit rate cannot exceed the readout rate of one super pixel. This rate is severely limited by the synchronous token ring which was chosen as the arbitration due to its simplicity. Regardless of the hit occupancy inside a double column, it takes at least 64 clock cycles for the super pixel to reacquire the token after it has released the token. When assuming an average rate of 1.2 kHz per pixel, this is not an issue. Because in real applications the occupancy may not be uniform, there may be local hot spots which produce data at a much higher rate. The maximum rate for a super pixel is approximately 400 kHz, and this rate is split between 8 pixels, meaning a rate no larger than 50 kHz per pixel can be sustained at local level. Because there is no additional buffering at the pixel-level, when the input hit rate in the pixel approaches the output readout rate, the efficiency drops quickly to 50 % as was discussed in Chapter 2.

When the hits are uniformly distributed across the double column, the maximum achievable rate from one double column is 1.5 MHz in the typical process corner. In this case, the architectural bottleneck is the handshaking between a super pixel and an EoC block, and the latency of the token contributes approximately 1/20 of the total latency only. By using a node-based data fabric, the rate could be increased to more than 6 MHz per column, when assuming 2 clock cycles for handshaking between nodes and 4 clock cycles for the data transfer (4×10 bits). This would increase the latency of the transfer to 4 clock cycles per each node on the transfer path. For example, data from the super pixel 63 would have a minimum latency of $64 \times 4 = 256$ clock cycles.

5.12 Concluding remarks

This chapter presented an HPD readout ASIC Timepix3, a successor to the Timepix HPD readout ASIC. Timepix3 introduced several new features when compared to existing readout ASICs. The architecture of Timepix3 offered simultaneous time and charge measurement capabilities with a pixel pitch of 55μm. It also offered a throughput of almost 80 Mhits/s per chip in a trigger-less readout mode. The chip also incorporated a simultaneous event counting and integral ToT mode.

Measurements and simulations are in agreement about the performance of the architecture. The chip has been manufactured, and has been observed to be fully operational in silicon, and to operate in adherence to the simulations.

Chapter 6

VeloPix ASIC

6.1 Motivation and requirements

The LHCb experiment at CERN will undergo electronics upgrades in the coming years [10]. In particular, the electronics of the VELO detector [9] of LHCb will be upgraded to handle higher hit rates. The VeloPix pixel readout ASIC is being designed to address this issue of unprecedented data rates which is a result of two major changes to increase the physics output of the experiment. The first change is a factor of 5 increase in the intensity of the proton-proton collisions and hence many more particle tracks producing hits in the pixel detectors. The second is the requirement to transmit the hit information from all collisions and not apply a hardware trigger filter.

Before delving into the architectural details and the performance of the ASIC, a short overview of the context where VeloPix will be deployed is presented. An artistic representation of the upgraded VELO detector is shown in Figure 6.1. The detector will contain 624 VeloPix ASICs in 26 sensor planes. It is a forward tracking detector with the closest chips being only 5.1 mm from the particle beam. The chips with the highest data rates are expected to see 8.5 particle tracks on average at a rate of 40 MHz [29]. Each of these tracks can create a cluster of hits producing event information in multiple pixels. As discussed later, this proximity of multiple hit pixels is used to reduce the overall data rate produced by the chip. The data rate handled by the hottest chips will be greater than in any previous pixel ASIC at CERN, and the chip has to transmit all events, without having access to a trigger signal for hit filtering, with a readout efficiency higher than 99 %.

A module of 12 VeloPix ASICs is shown on the left side of Figure 6.2. Each plane shown in Figure 6.1 consists of two of these modules. There are three ASICs per each sensor tile, and two sensors per each side of the module. The pixels in sensors covering the gaps between two adjacent ASIC are larger in area than rest of the pixels. The module has two sides, and the

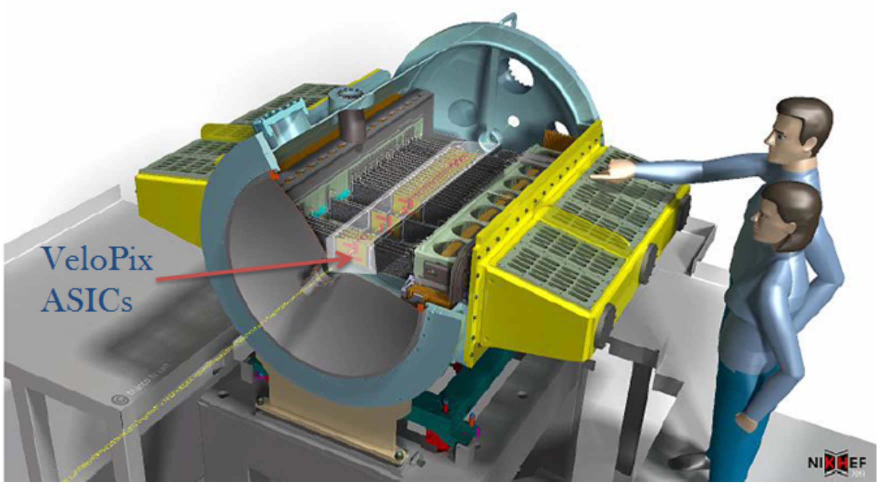


Figure 6.1: An artistic impression of the upgraded VELO detector containing 624 VeloPix ASICs. [29]

substrate is shown as transparent to show the ASICs on the backside, and the ASICs on the front-side are covered by the sensor tiles. The right side of this figure shows the track rates per ASICs seen by the hottest module. The radiation seen by different chips is very non-uniform, and this also results in a lot of variation in pixel occupancies.

Table 6.1 shows the requirements for VeloPix. The pixel size and the number of channels are identical to the Timepix3 ASIC presented in Chapter 5. This also conforms to the pixel size of other Medipix family chips [43, 25, 42], thus making it possible to use the same sensors with VeloPix already tested with these chips. The matrix operates with a 40 MHz clock which is also the frequency of the proton-proton collisions in the LHC. The chip will only record binary hit information from each pixel to minimize the data rate, and will time-stamp each hit with a precision of 25 ns. One orbit of the LHC is 3564 clock cycles or bunch crossings, and a 9-bit time stamp is used to identify one of these cycles by keeping the latency of the time stamped packets below 512 clock cycles. In this chapter, the time stamp will also be referred to as BX-ID.

One of the critical constraints of the design is the power consumption due to limited cooling possibilities. One of the objectives of the new module is to minimize the mass in the detector volume which prevents the use of large heat sinks for heat transportation. A micro-channel CO_2 cooling [107], also shown in Figure 6.2, will be used to prevent thermal runaway and to keep the chips at a temperature of around -20 C°, while minimising the material. The hit rates impose dead-time constraints on the design of the analog front-end, and on the bandwidth requirements for design of the readout

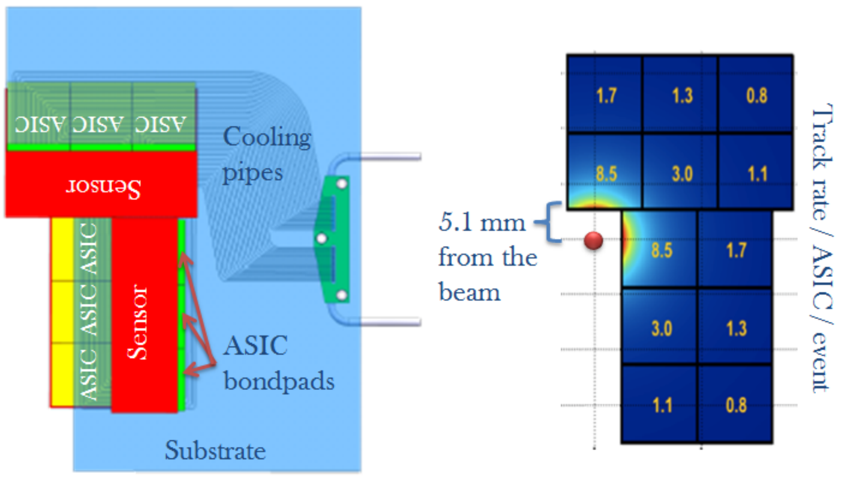


Figure 6.2: A module of 12 VeloPix ASICs designed for the VELO upgrade. [29]

architecture. A total bandwidth of 20.48 Gbps will be used to meet all the readout requirements above.

6.2 Architecture overview

A novel architecture for the VeloPix chip has been defined, simulated and implemented as part of this thesis to address the challenge of the high data rate and the restricted power budget. This architecture is a zero-suppressed, trigger-less, continuously operating packet-based architecture according to the definitions presented in Chapter 2. The architecture can operate with a duty cycle of 100 % thus being constantly sensitive to hits. In the following sections, the key parts of this architecture are discussed, and simulation results of the architectural performance are presented and compared to the requirements. The chip has not yet been manufactured in silicon.

The transport of the large volume of data produced by VeloPix has been investigated and then optimized based on the principles presented in previous chapters 2, 3 and 4. Timepix3, as presented in Chapter 5 was designed prior to VeloPix, and experience gained from that design is also used here. More precisely, issues found during the design and testing of Timepix3 such as digital-to-analog coupling of the time stamp bus, high packet latency due to column bus wire delays and hand-shaking protocol, and relatively low throughput per column have all been addressed in the design of VeloPix. Apart from the coupling, none of these were issues in Timepix3 due to longer ToA range (14 bits) and lower overall hit requirement

Table 6.1: VeloPix requirements.

Pixel size	55 $\mu m \times$ 55 μm
Number of channels	256 $\times$ 256
Operating frequency	40 MHz
ToT range	No ToT, binary
ToA range	9 bits
Minimum time resolution	25 ns (40 MHz)
Power consumption	$<$ 1.5 W/cm^2
Hit rate per pixel	50 kHz (peak) 10 kHz (average)
Maximum hit rate	460 Mhits/s/cm^2
Output bandwidth	20.48 Gbps (SLVS)

(40 Mhits/s/cm^2), but they must be addressed in the design of VeloPix.

6.3 Front-end description

The block diagram of the front-end including the super pixel logic is shown in Figure 6.3. The analog front-end processes current pulses arriving from the sensor input pad by amplifying the signal and comparing the output voltage of the amplifier against a programmed threshold value. The digital front-end processes the discriminator output by calculating the ToT value of the pulse and then comparing that value to a preprogrammed digital threshold. A valid signal is passed to the super pixel logic if the ToT value exceeds the threshold. This is done to reduce the effects of time-walk, and remove hits from small charges that could cause an extra packet to be created in the next clock cycle. Each super pixel logic module is connected to 8 digital front-ends.

6.3.1 Analog front-end

The analog front-end consists of an amplifier with a feedback capacitor of 3 fF implemented as a finger capacitor, a leakage current compensation and a discriminator. Leakage compensation is required to sink the constant leakage current coming from the sensor. Without the compensation, this current causes a DC offset which can move the amplifier out of its operating range. There is also functionality for test pulse injection which is digitally controlled. The expected detector capacitance C_{det} is approximately 50 fF. The front-end has a 4-bit DAC for threshold tuning, and these bits are set locally per pixel.

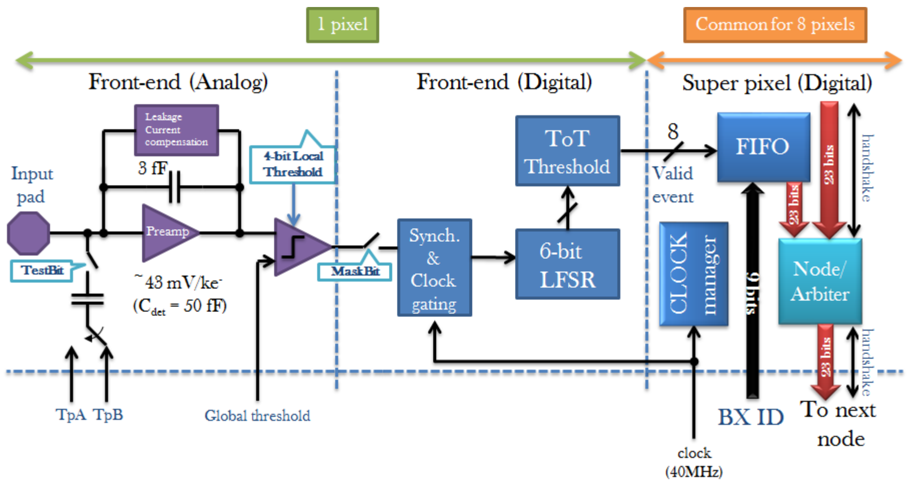


Figure 6.3: The block diagram of the front-end including the super pixel logic.

6.3.2 Digital front-end

The digital front-end has a synchronization logic for the asynchronous discriminator output, and a clock gating logic to reduce the power consumption of the digital front-end. A clock gating is especially important because the full chip contains over 65k pixels, and each logic gate and flip-flop consuming power is multiplied by this total number of pixels. As an example, one ungated flip-flop in 130 nm CMOS technology consumes 1 μW even when not changing its state. Thus preventing the propagation of the clock to one flip-flop in each pixel when that pixel is idle decreases the dynamic power consumption of the chip by 65 mW. Also, based on this figure, it can be concluded that the majority of the flip-flops have to be clock gated most of the time to reach the targeted power budget of 1.5 W/cm^2.

There is also a 6-bit LFSR per pixel for computing the ToT value of a hit, and a comparator for comparing the LFSR output to a preprogrammed digital value. This value can be programmed separately for each super pixel block of 8 pixels. The digital front-end also contains logic to set the threshold voltage for the discriminator via a threshold DAC. The LFSR can also be used to shift configuration data into the pixels, to read the data back or to read the ToT values from the chip. Under normal data acquisition operations, ToT values are not included in the output data, and are only used internally in the chip for vetoing time-walked hits.

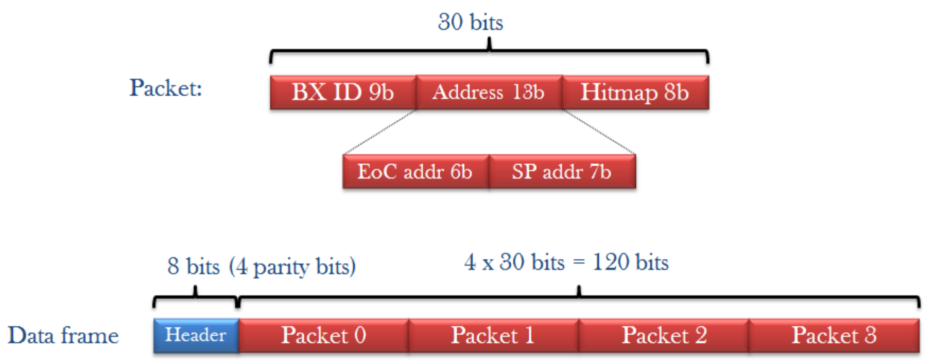


Figure 6.4: The data packet and the output packet frame of VeloPix.

6.4 Super pixel

6.4.1 Super pixel architecture

The main function of a super pixel block is to store the current BX-ID as a time-stamp if any of the 8 digital front-ends sends a hit signal. The state of all 8 pixels is also stored by the rising edge of this hit signal. An 8-bit OR of the digital front-end outputs functions as a latch-enable signal. Each pixel has a mask bit to set its output to 0 in case of a noisy pixel. The super pixel will add its address information to these data, and then send the packet down the column using a 1-D network of data nodes.

The full data packet of the chip is shown in Figure 6.4 along with the output data frame containing 4 packets. The packet and the framing scheme have been implemented to be robust against SEU by having fixed-length packets always at the same position in the frame. Each packet contains 30 bits of information including the time stamp, full address information of a super pixel and a hitmap of 8 pixels belonging to that super pixel. In addition to 4 packets, the frame contains a header for synchronization and a parity bit for each packet.

The super pixel has 5 logical blocks: a hitmap buffer, a clock manager, a data node, an arbiter and configuration logic. Each of these blocks is briefly described in the following paragraphs.

The hitmap buffer stores up to two super pixel packets. When at least one of the outputs from digital front-ends is high (an 8-bit OR), a packet is written into this buffer containing all information described above. The buffer has no dead time if it was the first packet written into it, and can accept a new packet on the following clock cycle. If the buffer is full, new information will be discarded and not recorded anywhere. To optimize the buffer for area, a linear FIFO is used, in which there are no input or output muxes, and a one bit control register per packet slot is required. Other features of the linear FIFO were already described in Chapter 4. For small FIFO depths, the additional latency and the number of control registers required are negligible. The routing of this FIFO is also simpler than with a FIFO equipped with muxes and read- and write-pointers because writes are always done to the same register, and the reads are done from the same register.

The arbiter has been implemented using a binary counter of 6 bits. In the case of a conflict for access to the data node, when the counter has a value 0, the hitmap buffer will be chosen as a data source for the data node. Otherwise, the previous data node is chosen as a data source. The counter is modulo 63 − *super pixel address*, and is thus slightly different in each super pixel. There is also a failsafe mechanism which clears the counter to 0 if an SEU flips the counter to higher than its modulo value. In this way, an SEU will only have a minor, transient impact on the arbitration priorities but the correct functionality of the logic is not compromised.

The clock manager in the super pixel serves two purposes. Firstly, it triplicates global reset and clock signals making the local routing robust to SETs. Because both of these signals are edge sensitive, any transient caused by SET can cause timing error or unintentional reset of the state registers. With the triplication, any state machine will recover on the next clock cycle as long as only one of the signals is upset. Secondly, it reduces the dynamic power consumed by the super pixel. The manager controls the clock gating by switching off the clock propagation to the super pixel module in the absence of any activity. The condition for enabling the clock is any of the following:

- Any pixel has its rising edge output high.
- The hitmap buffer has any packets in it.
- If the previous super pixel has any data in its node.
- If the data node has a packet to send.
- A read-operation between data nodes is in progress.

Figure 6.5 shows the comparison of the power consumption of the super pixel with and without custom clock gating. This custom logic has been hand-crafted because the compiler could not extract the conditions above. In [108], some techniques for refactoring the enable-conditions are presented to make the clock gating conditions recognizable to the compiler. Up to 50 % decrease in dynamic power is reported, but this generally depends on the type of the design, and on the application. One simple technique presented

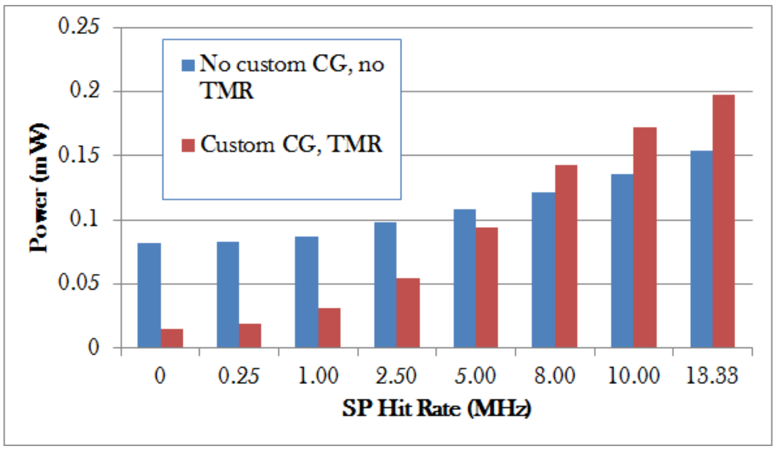


Figure 6.5: Power consumption of a super pixel with different hit rates.

there is to detect a change in the state register of FSM, and propagate the clock to the register only when the state is changing.

The targeted hit rate for a super pixel is approximately 0.3 MHz, and it can be seen in Figure 6.5 that with this rate the custom clock gating helps to reduce the power consumption. It is also notable that triplicated logic is included in the custom clock gating but not in the automatic one, and the custom clock gating consumes less power up to a hit rate of around 8 MHz. Additional techniques, such as data-driven clock gating [109], in which an XOR-gate is connected between inputs and outputs of a register and an OR-function of the output of the XOR-gate used as a clock enable, could also be employed to optimize the power consumption. However, generally there is very little correlation in the data between two successive super pixel packets making this clock gating technique ineffective for large register sizes. For smaller sizes, the overhead of having extra gates inserted cannot be afforded due to the restricted area available.

The data node handles all data communication external to the super pixel. Each node stores up to one data packet, and tries to pass it to the next node if the next node is empty. The interface between two nodes is based on a pull-architecture where the next node pulls a packet from a previous node. This happens only if the previous node has valid data, and if the next node has capacity to store the packet.

6.4.2 Choice of super pixel dimensions

The choice of the super pixel size can have significant impact on the data rate produced by the chip. The data reduction is achieved by sharing time stamp and address information between several pixels. The magnitude of

Table 6.2: Data rates with different super pixel geometries. N is log_2(pixels in super pixels).

SP geometry	Data rate (Gbps)	Packet size (#bits)
256×256	14.785	$28 + \text{N} \times 16$
4×4	16.477	$28 + \text{N} \times 4$
2×4	16.806	$28 + \text{N} \times 3$
$2 \times 4^*$	16.945	33
1×4	17.758	30
2×2	17.857	30
1×4	18.283	$28 + \text{N} \times 2$
2×2	18.373	$28 + \text{N} \times 2$
4×4	18.661	40
1×2	19.703	29
1×2	19.878	$28 + \text{N} \times 1$
single pixel	23.916	28

this reduction depends heavily on the cluster size, and can be even negative if most of the tracks consist of single pixel clusters.

Table 6.2 shows the data rates produced by different super pixel sizes. These numbers have been used by using algorithmic grouping techniques described in Chapter 3, and then forming the actual packets based on the groups obtained. These numbers were obtained using a 12-bit time stamp which was later optimized down to 9 bits. The left-hand side indicates the super pixel geometry, the middle column the data rate produced using that geometry and the right-hand side how the packet size is calculated. The N in the packet size indicates a number of pixels that were hit in that super pixel region. Note that these variable-sized architectures are more complex to implement as they need counters or other monitoring logic to keep track of the packet lengths. The first entry, a 256×256 super pixel is shown as an ideal reference in which all hits from one bunch crossing are captured under a single time stamp. In this case the packet will contain 12 bits for the time stamp, 16 bits for the number of pixels hit and a 16-bit address for each pixel hit. As the last entry, a single pixel case is shown where each pixel captures a time stamp on its own. The chosen solution, marked with *, has a data reduction of 30 % compared to a single pixel case.

So far, these kinds of encoding techniques for data reduction have not been widely employed in pixel readout chips. A 2×2 super pixel has been used in [75] using a fixed packet length, also adding ToT information for each hit. They report a data reduction of approximately 15 %. Although the choice of a 2×4 super pixel architecture seems to result in 30 % reduction of data, the 4×4 super pixel or even larger dimension would seem to offer even more reduction. In this case however, the choice is also dictated by the implementation details.

Both the 2×4 and 4×4 architectures showed a similar reduction in data rate. However, the 4×4 super-pixel and also larger dimensions were rejected after looking at the implementation details. To have identical routing from the input pads to the analog front-ends requires symmetrical layout. This symmetry minimizes the mismatch between different pixels. In a 4×4 or larger architectures, analog islands would be required to support this kind of symmetry. The trade-offs and issues related to 4×4 layout and the analog islands were already discussed in Chapter 4.

There are other means for data reduction, for example, lossless compression techniques like run-length encoding and Huffman encoding [110]. One drawback of these techniques is that symbols in the encoded data stream depend on each other, and if one of the symbols gets corrupted by an SEU, all following symbols are corrupted as well. Thus, their usage would require ECCs to ensure the correctness of data. Another property of these techniques is that their compression efficiency decreases when the entropy or randomness of the data increases. To determine the entropy of a stream of data, a formula developed by Shannon [111] is used:

$$H = -\sum_{i=0}^{n} P(n) * log_2(P(n)) \tag{6.1}$$

where H is the (Shannon) entropy of n different symbols and $P(n)$ is the probability of a specific symbol appearing in a data stream. As an example, if the Equation 6.1 is applied to a byte stream, H will indicate how many bits are required per byte to encode the data stream with an ideal, lossless compression. For practical algorithms like the run-length encoding and the Huffman encoding, this number will be higher. If Equation 6.1 is applied to a data stream from VeloPix, it can be seen from Figure 6.6 that the entropy is high in relation to the symbol size. The calculations were done using 20.7 M packets from VeloPix output data which means that no sufficient amount of statistics is available for symbols greater than 24 bits.

It should be understood though, that H is above all a measure of entropy of RNGs used in the simulations as those are the only source of randomness present in the simulation. Assuming that the RNGs are good representative of the actual application environment for VeloPix, it can be concluded that no efficient lossless compression algorithm exists which could be used to further decrease the data rate.

In simulations related to the VELO upgrade and track reconstruction [112] it is also mentioned, that the processing of clusters could be made two times faster by including an additional flag in the data packets. This flag would indicate that a region of 2×4 pixels had no hits in any of the 8 neigh-

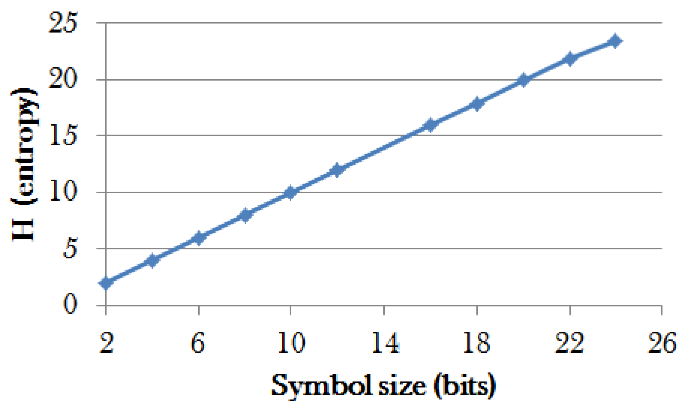


Figure 6.6: Entropy of the data from VeloPix.

boring super pixel. This functionality has not been implemented because it requires additional routing resources between the super pixels, and it is not a crucial optimization for the tracking performance. Also, if done within the ASIC, no cluster split between two ASICs could be detected in this manner. Recall that each sensor in the upgraded VELO will be connected to 3 separate ASICs by bump-bonding which can result in a cluster of hits being split between two readout ASICs. Detecting these split clusters has to be done in a system receiving data from three separate ASICs.

6.5 Column readout architecture

The column readout architecture of VeloPix is shown in Figure 6.7. The architecture uses a scheme presented in Section 2.13 where the arbitration of the node access between the local buffer (a super pixel hitmap buffer) and the previous node is done using weighted round-robin (WRR) scheme. As previously described, this scheme assigns a weighted priority for each local buffer which depends on the total numbers of nodes and the address of the local buffer. It can be seen that the only global signals utilized are reset and clock. The communication distance between two nodes is at most 220 μm, which equals 4×55 μm pixels, so meeting the timing constraints is easier than when using a synchronous bus spanning the full column.

Communication between nodes is done by first asserting a valid-signal if a node has received data from a local buffer or from a previous node. The next node in the chain then asserts its read-signal to notify the previous node of the acceptance of the packet. After the transmitting node registers that the read-signal has been asserted, it deasserts its valid-signal, and the receiving node then deasserts its read-signal. This same sequence is per-

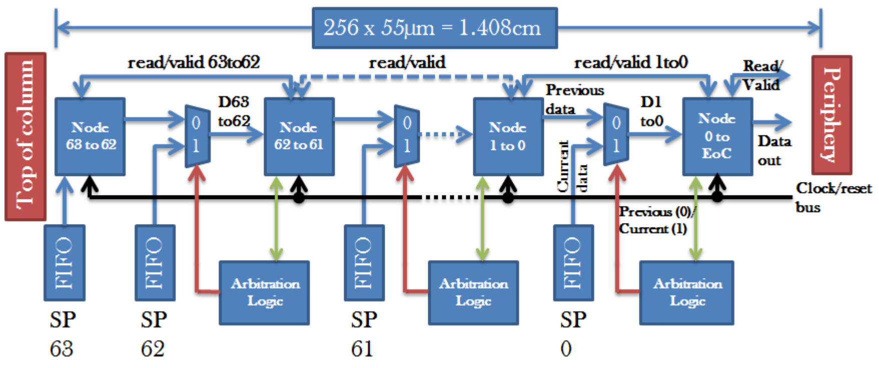


Figure 6.7: A block diagram of a column architecture of VeloPix.

formed between all neighboring nodes until the packet reaches the node 0, and ultimately the periphery logic. In effect, this kind of communication results in latency proportional to the physical location of the original source of the packet.

To make meeting the timing easier, a fully asynchronous approach could also be used. In [113], an SEU robust approach is presented for asynchronous pipelined communication links. The disadvantage of this approach to the fully synchronous one adopted for VeloPix is the hardware overhead. Such an asynchronous approach requires twice the number of wires if using latency-insensitive protocol and dual-rail encoding. Note that the data must be fully protected because the completion of the transaction is encoded in the data itself. In the fully synchronous communication between nodes, only the control signals need to be triplicated to ensure the correctness of communication between the data nodes. Another advantage of the asynchronous approach is obviously that the throughput is not determined by the slowest timing path which determines the maximum clock frequency.

6.6 End-of-Column and Periphery

The first part of the periphery data path is shown in Figure 6.8. A novel feature of this architecture is that it has been split into two sides. Each side has four data fabrics, each consisting of 16 nodes. The nodes have been connected such that every fourth EoC connects to the same data fabric per side. This has an effect of distributing the packet traffic more evenly between the data fabrics. Each fabric is connected to the center node which arbitrates between channels arriving from left and right. A left-right channel pair is matched to one of the four output channels. The matching is static, and if one of the pairs has no valid data, the corresponding output is also left unused. This happens even in the case where multiple other pairs may

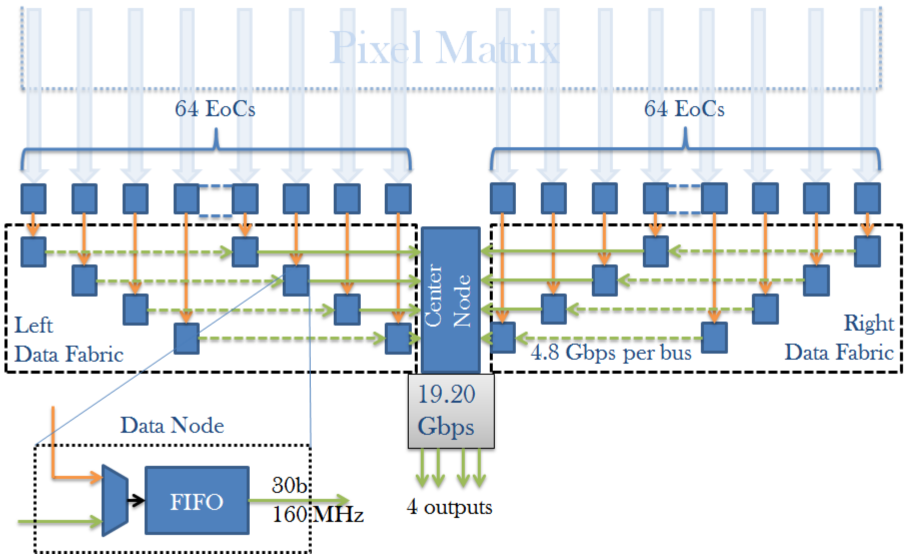


Figure 6.8: The periphery data path of the VeloPix showing 128 EoCs and the data fabric.

have congestion. This is done to simplify the logic instead of always trying to match exactly 4 available inputs to the 4 outputs. The simplification is also done because the topology of the EoC connections to the fabric already distribute the traffic very evenly.

This data fabric is similar to the column architecture but each node contains a FIFO instead of a single register. This makes it possible to send and receive a packet every clock cycle without blocking. The arbitration for each node is done as in the column by using a binary counter and selecting the local buffer, in this case EoC, instead of the previous node. The difference in this case is that each arbitration counter counts up to its node address on the left side, and up to $15 - \textit{node address}$ on the right side. The leftmost node in each of the data fabric channels has an address of 0 and the rightmost one an address of 15.

The second part of the periphery is shown in Figure 6.9. Inputs shown on the left of the figure are connected to the 4 outputs of Figure 6.8. The first block is 4×4 cross-bar switch which is used to map any of the input channels to any of its outputs. The benefit of this is that any channel can be turned off to reduce the power consumption if the full bandwidth need not be utilized. For debugging purposes, a 128-bit readout register is connected to the router. All data can be read out through an 80 Mbps serial link instead of using the faster 5.12 Gbps links.

After routing, data in each packet is scrambled for better DC-properties. Then after scrambling, packets are framed into 128-bit frames containing 4

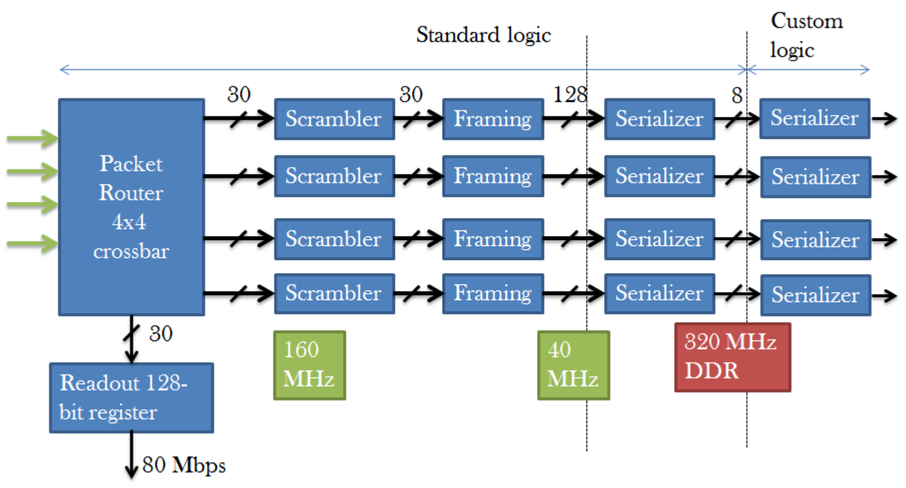


Figure 6.9: The output logic of the VeloPix periphery.

data packets of 30 bits. These frames contain a 4-bit fixed header for frame synchronization at the receiver side and a 4-bit code containing the parity bits of each packet. These frames are then converted into 8-bit words in the following serialiser. The final serialiser performs 8-to-1 bit serialization. 8 bits are loaded at 320 MHz DDR, and this results in a serial stream of 5.12 Gbps. More details and the architecture of this serialiser can be found from [114].

6.7 Monte Carlo data sets and hit generation

Results shown in the following sections are partly based on the MC physics simulations of the VELO upgrade. The MC simulation methods of the VELO upgrade are explained in [112]. At first, the event data is produced by using GAUSS [115] application which produces the primary events. As mentioned in [112], this part of the simulation does not require any VELO specific parts. The second application, BOOLE, maps the generated events to a detector of specific geometry, and produces the coordinates of the actual pixels. This output data from BOOLE is used as an input to the architectural simulations.

Due to the module structure of 12 VeloPix ASICs presented before, the hit occupancy in the hottest chips is highly non-uniform. These distributions were extracted from the MC data, then these data were formatted into packets using an algorithm to pack hits into 2×4 super pixel packets. The address distributions of these packets were recorded, and a weighted RNGs in C++ and SystemVerilog created out of these.

6.8 Architectural simulation

This section summarizes the most important simulations related to the architectural performance of VeloPix. As stated in the introductory section, a readout efficiency higher than 99 % for a full chip is targeted. It will be seen that under non-uniform occupancy, some parts of the architecture may locally have efficiency lower than 99 % while the overall efficiency is still higher than this limit.

It was already shown in Chapter 3 in Section 3.2.8 that high-level simulation models can be used instead of RTL models to reduce the run-time of simulations considerably (more than 20×) and to increase the number of cycles simulated for collecting more statistics. Particularly, it was shown for VeloPix models, that a reduction of two in LoC was achieved using sequential high-level model.

6.8.1 Front-end pile-up

The efficiency of the analog front-end is shown in Figure 6.10. The plot has been obtained by creating a high-level model of the front-end based on characteristics obtained from a transistor-level simulation. The preparation of the model and the simulations were carried out for this thesis. The model has been created using a LUT technique described in Chapter 3. It can be seen that the losses in the hottest region of the chip are up to 1.6 %. The loss occurs every time a front-end is still processing a hit while another hit arrives. This hit also extends the dead-time of the front-end to model the charge pile-up, so it is a paralysable front-end architecture [60]. In the high-level model, a counter is loaded with a value proportional to the arriving input charge, and decremented by one during each simulation cycle. Thus, pile-up occurs every time a hit arrives and this counter is not 0.

6.8.2 Readout efficiency

The readout efficiency of the architecture was measured using different input packets rates. It is shown in Figure 6.11 together with the ideal response having no super pixel pile-up, and infinite time stamp range. No analog front-end pile-up is taken into account in this figure. It can be seen that the linear super pixel pile-up region extends up to a rate of 600 Mpackets/s. The rate of the efficiency loss beyond that could be reduced by increasing the buffering capability of super pixels in the architecture chosen for the VeloPix. After this linear region, starting from an input rate of 600 Mpackets/s, losses start occurring due to too high a latency. The losses start increasing more rapidly after this point, and can be decreased only by extending the time stamp range or increasing the output bandwidth. An extreme simulation case with an input rate of 1.28 Gpackets/s shows that

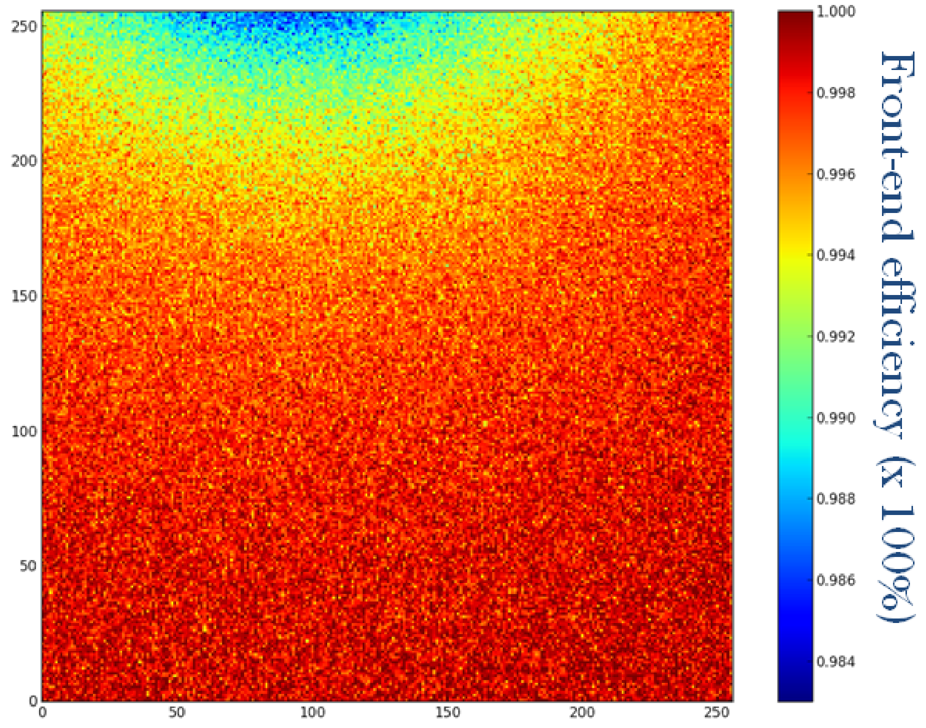


Figure 6.10: Efficiency of the analog front-end.

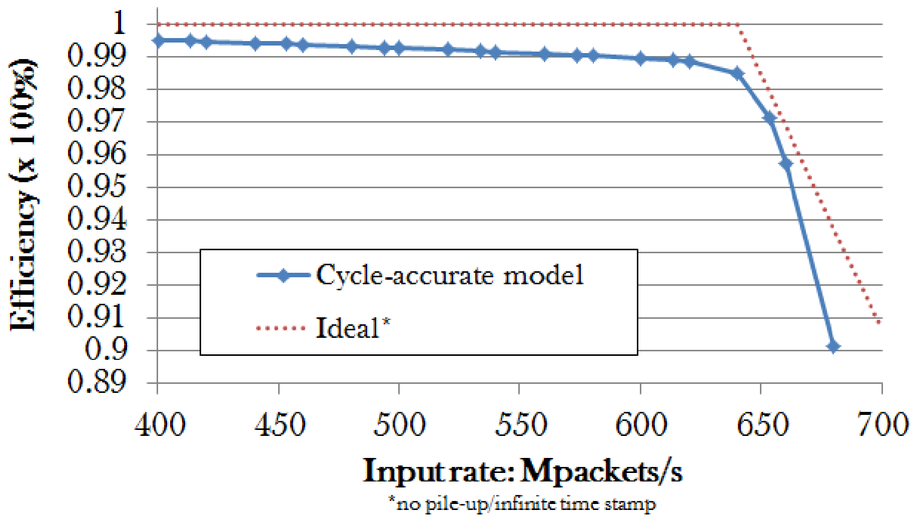


Figure 6.11: Readout efficiency of the VeloPix architecture with different input rates.

the readout efficiency drops below 10% because most of the packets come out with a latency exceeding the time stamp range of 9 bits.

6.8.3 Latency

The time stamp range for VeloPix was chosen to be 9 bits recorded at 40 MHz. This range was extracted from the simulations by changing the input rate, and measuring the efficiency with different time stamp ranges. It was assumed, that if the latency in a packet exceeded the time stamp range, that packet would be marked as lost. The latency with different input packet rates is shown in Figure 6.12. Due to the column readout architecture and the non-uniform distribution of hits, the most frequent value is around 64. This is caused by the fact that there are 64 nodes in one column, and the rate is highest in the node at the top-level.

It can be seen from Figure 6.12 that the 9-bit time stamp range is sufficient for the targeted rate which is less than $< 540 Mpackets/s$.

6.8.4 Model validation

High-level simulation techniques used in this thesis for pixel chips were described in Chapter 3. The benefits of these models were already described including shorter run-time and easier implementation, and a non-synthesizable model was also used for VeloPix to validate the performance of the archi-

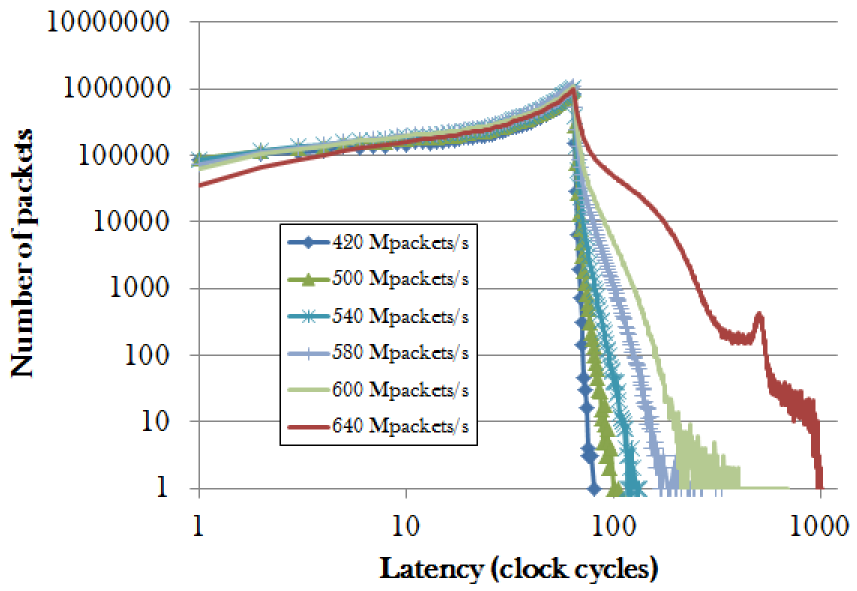


Figure 6.12: Latency of the VeloPix architecture with different input rates.

tecture. After an architecture was chosen and implemented in RTL, the performance of the RTL model was validated to conform to that of the high-level model. The cycle-accurate simulation was run for 1.6 Mcycles, the RTL simulation for 0.8 Mcycles and a packet rate of 520 Mpackets/s was used. This is the expected peak rate of the hottest chips in the upgraded VELO based on the MC physics simulation data. The running of the RTL simulation took 36786 s which equals to a simulation speed of 0.3 kpackets/s. The cycle-accurate model took 583 s equaling simulation speed of 36.7 kpackets/s. The readout efficiency in the RTL simulations was 99.6% with a measured output rate of 518 Mpackets/s and it was 99.2% (515 Mpackets/s) in the cycle-accurate simulations.

The latency of the packets from both the simulations is plotted in Figure 6.13. Based on the plot, and the similarity of readout efficiency values, it can be concluded that the cycle-accurate model and RTL model are in agreement with respect to the simulation results. This is an important validation to do after the actual circuit description has been done, either in RTL or in a full-custom manner.

6.9 Post-layout analysis

In this thesis, the column architecture was also verified after the PnR, and metrics such as power, area and timing were measured. The analysis was

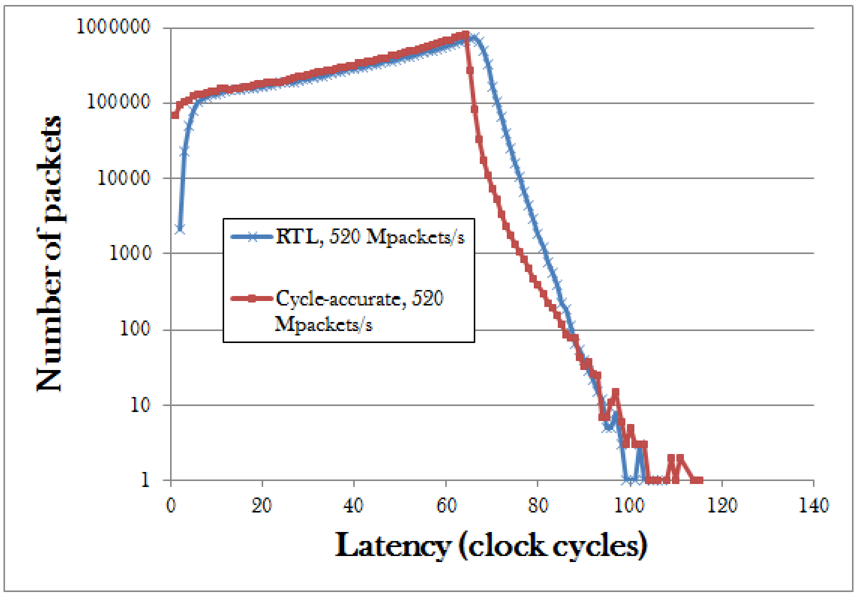


Figure 6.13: Latency of the two chip models with targeted input rate.

done at a double-column level as timing, power and area in one column are independent of other columns. A partial floorplan of the column is shown in Figure 6.14. It shows that the structure of a super pixel is 2×4 pixels, split into analog and digital front-ends, and the super pixel logic area. Only signals related to data transmission are shown in the figure, and the configuration signals are omitted. The signals traversing up the column are buffered every 880 μm. The event data is transported from one data node register to the next, and this distance is always less than 220 μm. An important consideration was the replacement of the BX-ID bus more towards the center of the column and the digital logic. In Timepix3, this bus was routed under the bump pads but in VeloPix it has been moved to the center of the double column. Because the bus need not be distributed into the digital front-ends, this relocation does not consume horizontal routing resources.

6.9.1 Power consumption

As the VeloPix is targeted for an application with a relatively high duty cycle of > 66 %, minimization of activity-related dynamic power consumption is the first priority in power optimization for the digital architecture. Firstly, an estimate of the blocks consuming most of the power must be obtained, and then secondly, corresponding power optimizations to minimize consumption in these parts must be implemented.

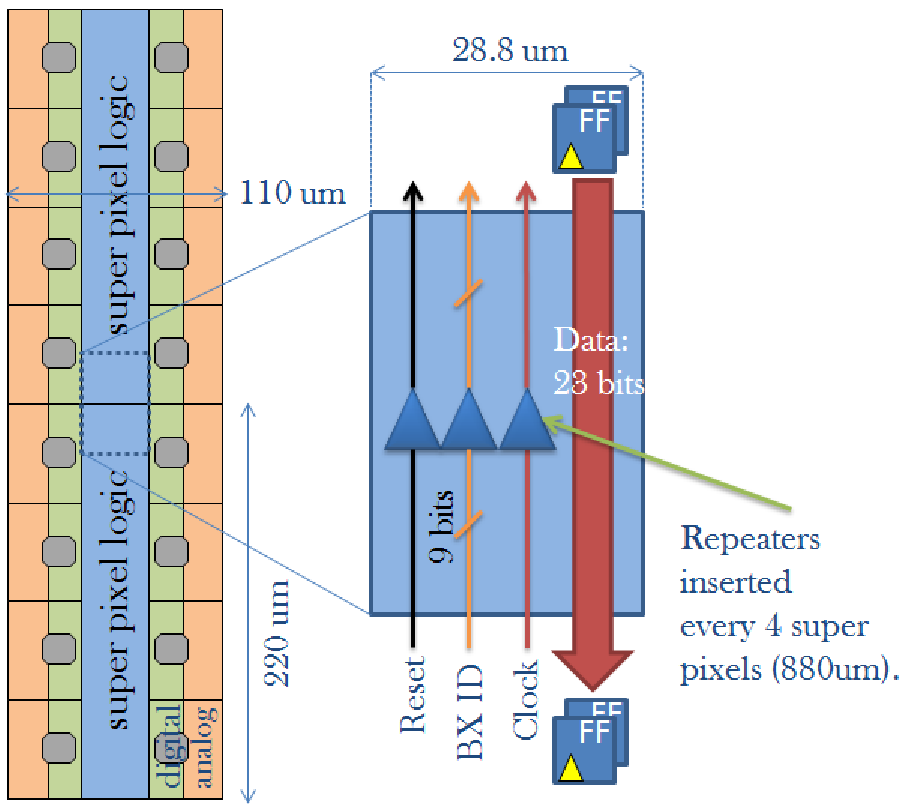


Figure 6.14: A floorplan (2 super pixels) of the column of VeloPix with global signals.

An estimate for the power consumption of the digital blocks was obtained by running digital simulations with a post-layout netlist and corresponding standard delay format (SDF) file. From these simulations, a VCD file of the digital activity was created, and imported back to the PnR tool. Based on the activity in the VCD file, the power calculation engine of the PnR tool calculates dynamic power consumed by each digital cell. All power figures have been obtained using a fast process corner, a power supply of 1.6 V and a temperature of $-55\ C^{\circ}$.

As shown in Figure 6.5, the power consumption of a single super pixel was first obtained, and the super pixel characterized with different rates of activity. After this, based on the activity per column obtained from RTL simulations, power consumption for a full column was obtained. The power of this column was scaled with the activity obtained for all columns. Power consumption independent of the data activity, like global signal distribution and power consumed in an idle state with no data to process, were also added to the final estimate.

To reduce the leakage power of the chip, high-Vt transistors were used in the library cells deployed in the pixel region. It was estimated from the output of the synthesis tool, that the total leakage for one super pixel block was less than 12 nW, at temperature of 125 C° and voltage of 1.4 V. From this number it can be concluded that the leakage power is not an issue if the chip is kept under this temperature.

Power consumption for a clock tree per column was estimated to be 476 μW, and 273 μW for the BX-ID distribution. Note that the dynamic power is independent of the number of bits in the BX-ID bus because Gray encoding is used. The estimation was made by placing and routing a full column of 64 super pixels using an identical layout for all super pixels, and inserting repeaters for the clock and the BX-ID between the super pixels in a manner shown in Figure 6.14.

Activity-related power consumption of the digital data path in the pixel matrix (excluding digital front-ends) is shown in Figure 6.15. This is derived from the hit patterns produced by the MC simulations of the VELO. These numbers do not include power consumed by the logic in the absence of any hit-related activity, because the power consumed in this idle-state is the same for all super pixels. The activity-related consumption is directly proportional to the packet rates in these columns. The spike in the last column (127) is due to the bigger sensor pixels connected to this column which detect a higher rate of hits because of their size.

6.9.2 Timing

Although VeloPix is foreseen to operate with a 40 MHz clock in the pixel matrix region, the maximum clock frequency for the design was determined.

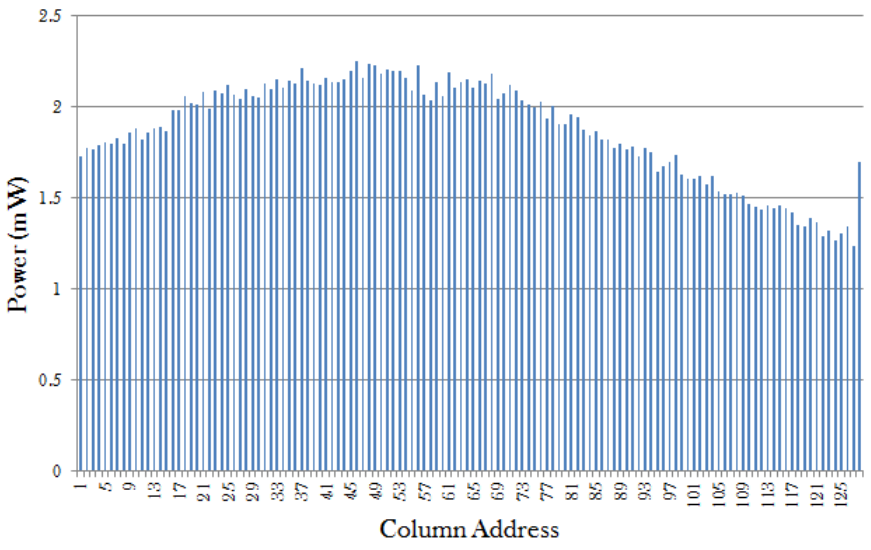


Figure 6.15: Digital power consumption of the pixel matrix per column.

This was done to study the scalability of the system for applications demanding even higher rates, 1 GHz/cm^2 and beyond. As shown in Figure 6.14, the area allocated for the super pixel logic is 28×110 μm, and it was kept constant for the timing studies.

The column was successfully placed and routed at 100 MHz. This would bring the total packet rate delivered by the pixel matrix to more than 4.2 Gpackets/s. Reading out packets at this rate would require an output bandwidth of more than 120 Gbps. This in turn would require 24 output links running at 5.12 Gbps. Assuming a power consumption of 60 mW per link [114], the total power of the links would exceed 1.4 W. In addition to this, the increase in dynamic power consumption due to increase in the clock frequency has to be taken into account. Thus, it can be concluded that the timing in logic would permit an increase in the clock frequency, but there are other limitations such as power which do not permit increasing the packet rate by this kind of approach.

Other critical paths for the timing are the global signals (clock, reset and time stamp) distributed along the column, shown in Figure 6.14. The delays for each of these signals are shown in Table 6.3. The quoted numbers have been obtained using a slow simulation corner, temperature of 125 C° and a power supply of 1.4 V (SS/1.40V/125C°). The signals are transmitted to the column using the negative edge of the clock to prevent hold time violations in the first super pixels. Although the time stamp and the reset would not meet the timing for 100 MHz, this does not prevent increasing

Table 6.3: Timing of global signals in a column.

Signal	Skew (rise)	Skew (fall)
Clock	1670 ps	1830 ps
Reset	7130 ps	8540 ps
BX-ID	8110 ps	9950 ps

the readout frequency. The time stamping could still be performed using a 40 MHz clock derived from the distributed 100 MHz clock.

6.9.3 Single-event upset tolerance

Special design techniques must be adopted to protect the logic against functional errors due to SEUs. The basic techniques for improving the computer reliability by using TMR are presented in [23]. A number of other design techniques have also been described in literature for SEU mitigation of digital logic. In [116], a word-based voter is presented. This voter checks all voted outputs and reports an error if at least two outputs are not equal. This gives an extra safeguard against single-event multiple-bit upsets (MBUs). A TMR flip-flop is presented in [117] which embeds a voter and error detection circuitry. It was already concluded in the Chapter 1 that the area overhead for a full TMR is always more than 200 % compared to a non-TMR option. In addition to TMR, the so-called dual interlocked cell (DICE) [118] can be used to increase the tolerance of latches to SEUs. In these cells, each memory node has 4 cross-coupled inverters instead of 2, and an SEU in any node can be corrected automatically by the feedback logic in the DICE.

Due to its simplicity and ease of integration into the RTL design flow, a full TMR scheme was chosen in which all voters, logic and registers are triplicated on critical paths. The triplication was embedded into the RTL description of the logic to have a full control of the triplication. Because of the limited area in the pixel matrix, only FSMs, control and state registers such as FIFO state flip-flops are triplicated. Also, the configuration latches in the pixel matrix are triplicated, and automatically self-correcting latches are used in the pixel matrix. These latches do not use the DICE technique but are triplicated latches instead, which refresh themselves with a voted output of the three latches in the case any of the three outputs differs from the other two. The triplicated latches were found to be smaller in area than the DICE, and were already implemented in the standard cell library used for this work. There is no error correction for data within the data packets themselves. Only a parity bit will be transmitted off-chip but it cannot detect internal bit-flips that occurred before adding the parity bit. An error rate of 1 per 1000 packets is deemed to be acceptable, and is estimated to have no impact on the overall tracking efficiency of the upgraded VELO.

A specific naming convention was adopted for triplicated nodes in the RTL code to indicate to the synthesis tools that these nets should be preserved. All these nets were labelled with a suffix `_tmr`. Using the SDC command set_dont_touch, the optimization of the triplicated nets was prevented.

An SEU generator was also used to verify the SEE tolerance of the logic, and the preservation of triplicated signals by the synthesis tool. The generator was produced by parsing post-layout netlists of the modules, and generating an SV module containing a `randcase`-statement including all registers, latches and nets in the design. This statement contains `force`- and `release`-statements to inject SETs into these nets, and simple inversion of latch and flip-flop values in the case of SEUs. The SEE verification had to be done at subblock-level only (for example the super pixel, the EoC block, the packet router) due to slow parsing of netlists, and due to large number of wires and instances in larger, top-level blocks. For each block, the generator was instantiated at the top-level testbench of that block along with other testbench logic, and post-layout simulations performed to ensure correct operation even in the presence of SEE. One shortcoming of this method was that no node capacitance was taken into account, and all nodes had an equal probability of being upset, and equal duration for SETs. This could be corrected by using a standard parasitic extraction format (SPEF) file to provide the node capacitance information for the parser.

6.10 Concluding remarks

A novel HPD readout ASIC architecture was presented in this chapter which will be incorporated into the VeloPix chip. This chip will be used as a readout chip of the VELO detector of LHCb after its electronics have been upgraded to handle higher luminosities. The architecture, which is also SEU tolerant, delivers up to 600 Mpackets/s with a readout efficiency greater than 99 % when using a 9-bit time stamp. It was also shown that the latency of data with this rate is below the range of the time stamp guaranteeing unambiguous off-line event identification.

In Chapter 3 it was seen that using high-levels models above the RTL abstraction can reduce the simulation time by orders of magnitude. In this chapter it was shown also that the accuracy of the simulation results is very close to the RTL model. Using high-level models was an essential part of the optimization process because parametrized simulations could be run in a matter of minutes instead of hours or days using high-level models. That being said, the performance of the chosen RTL architecture was nevertheless verified using a simulation run lasting for tens of hours.

Finally, critical metrics such as timing and power for the architecture were estimated. It was concluded that the column readout architecture would scale from 40 MHz to 100 MHz clock frequency thanks to the localized communication between data nodes but at the cost of increased power. Also, the increase in the packet throughput of the pixel matrix would have to be handled by other means than simply increasing the number of the output links.

Chapter 7

Conclusions and Future Work

The goal of this thesis was to find out how digital data transfer in a hybrid pixel readout chip could be improved in terms of absolute rates given a limited area, and how the power efficiency could be enhanced. Naturally, when more precise measurements and better image quality are desired in imaging applications, more data and thus higher output bandwidth are a consequence.

The thesis has discussed and presented various options of digital data transfer in the readout chips of HPDs taking into account area and power limitations present in these applications. Particularly, readout architectures for applications requiring a pixel pitch of 55 μm were presented. The presented techniques, which are especially adapted for area and power limitations of mixed-signal pixel readout ASICs, can be used to improve the readout efficiency of readout architectures. The techniques presented here are also compatible with modern digital ASIC design techniques and standard cell design flow making it possible to quickly port them into new CMOS technologies. As the complexity of HPD readout ASICs has also been increasing, these techniques are important in reducing the design times and the number of design bugs. As on-chip wire delays do not scale down similarly as gate delays when moving to newer CMOS technologies, the presented architectural techniques have also addressed this problem either by using asynchronous communication (Timepix3) and data-node based architecture (VeloPix).

7.1 Empirical findings

Following the design principles laid out in Chapters 1, 2 and 4 and analyzing the resulting readout architectures for the two readout chips presented in this thesis, namely Timepix3 and VeloPix, the following observations can be made:

- It was shown by post-layout simulations, analog simulations and a synthesis tool in Chapter 4 that the traditional bus-based architecture is not an optimal choice for the readout architecture of a pixel column in terms of throughput-to-power ratio. It was also shown that achieving a similar throughput for a bus as for a node-based data fabric is difficult in practice, especially with pixel pitches 55 μm or smaller.

- Despite the area constraints in the pixel matrix, it is possible to implement a network of nodes inside the pixel matrix, even with a pixel pitch of 55 μm. This technique must be combined with the super pixel grouping technique to fit the area, which has been proven in the implementation and functional success of Timepix3.

- An architectural solution to reading out 80 Mhits/s from a single readout chip was presented in Chapter 5, namely in the readout chip Timepix3. Simulations and tests showed that in terms of measurement accuracy, the presented solution was equal or better than the previous chips, while it improved the throughput, readout efficiency and offered simultaneous measurement of charge and time.

- A readout architecture delivering 640 Mpackets/s, up to 8 hits in each packet, was presented in Chapter 6. This architecture is used in the readout chip VeloPix, which will be used for the upgrade of the VELO detector at CERN. The architecture is tolerant against SEUs and delivers data from the pixel matrix using a data fabric based on locally communicating nodes. It can achieve 13.3 Mpackets/s per double column, with a column width of 110 μm. A similar approach is also followed at the periphery of the chip.

The bottlenecks in architectures have also been identified. Generally, the bottleneck is either found from the pixel-to-EoC data transport or at the output of the chip. Using techniques in this thesis, the former bottleneck can be easily eliminated. However, in future applications, when facing hit rates of multiple Ghits/s/cm^2, it may not be feasible to transmit this data off-chip, even if the internal architecture could sustain the rates. More intelligent and more efficient data reduction techniques are thus required. This is becoming a more common requirement in systems where triggering is no longer desired, as seen in the study for VeloPix.

The most relevant characteristics of Timepix3 and VeloPix are summarized in Table 7.1. It can be seen that Timepix3 offers much more precise

Table 7.1: Summary of features in Timepix3 and VeloPix.

Feature	Timepix3	VeloPix
Pixel size	$55\ \mu m \times\ 55\ \mu m$	$55\ \mu m \times\ 55\ \mu m$
Matrix size	256×256	256×256
Super pixel	2×4 pixels	2×4 pixels
Power	< 1 W/cm^2	< 1.5 W/cm^2
Time resol.	1.5625 ns, 18 bits	25 ns, 9 bits
Max. hit rate	40 Mhits/s/cm^2	> 320 Mhits/s/cm^2
Charge/ToT	10 bits @ 40 MHz	Binary
TID	N/A	Up to 400 Mrads
SEU tolerant	NO	Yes (simulated)
Data packet	48 bits (37 bits)	30 bits (23 bits)
Architecture	Asynch. bus	Synch. data fabric

charge and time measurement possibilities than VeloPix, while VeloPix focuses on delivering larger volume of data in higher particle fluxes. The packet length inside a column is shown in parentheses.

It can be concluded that an asynchronous bus is a robust architectural option for lower rates giving full separation of the clock domains between the pixel matrix and the periphery. This fact was exploited in Timepix3 by using 16 different clock phases in the pixel matrix to reduce current spikes due to the clock buffering. The bus-based architecture is also more suitable for split data transfers required for a higher number of bits per data packet because an RX module is only required at the EoC. For the data fabric, each node must contain RX and TX modules.

Synchronous data fabric is a high throughput solution for moderately sized data packets. The hardware implementation is especially simple if the full packet can be transferred in parallel. It is also useful for absorbing data rate fluctuations when the hit occupancy is non-uniform and focused on the top of the matrix. In this case, the nodes in the fabric act as a distributed data buffer.

7.2 Theoretical implications

This thesis has emphasized the importance of high-level simulations when choosing the architecture for an HPD readout ASIC in Chapter 3. The main reasons for using these high-level simulations are reduced simulation run-time, design and debugging effort and the ability to run simulations for long enough to reduce the impact of initial transients and measurements fluctuations.

Another implication of this study is to move away from bus-based architectures to improve the power efficiency of the readout architectures in hybrid pixel chips. Node-based fabrics and networks are scalable with respect to the size of the pixel column because their performance does not suffer from increased wire delays in newer technologies. In fact, closely placed data nodes can be run at even higher frequencies in new technologies because the speed of the logic improves due to CMOS scaling.

7.3 Limitations of study

This study has not addressed fully asynchronous, clock-less techniques which could be used inside the pixel matrix. The main reason for this is that a clock signal is typically present in tracking applications in which timing measurements are required. This means that because the clock is already present due to the measurement requirements, it can also be used for readout purposes without extra power or area. Another reason is that protection against soft errors like SEEs is more costly in asynchronous logic in terms of area. On the other hand, very few chips [63] have used completely clock-less readout architectures.

This study also has not investigated thoroughly the digital-to-analog coupling effects which occur in the pixel matrix either via the power supply noise or through capacitive coupling from switching digital signals. These effects have been clearly observed in the manufactured chip Timepix3, but at the time of writing, the effects have not been reproduced fully in simulations.

The architectural techniques presented here also have their limitations with respect to the minimum pixel size. Given the same CMOS technology that was used in this study, namely 130 nm CMOS, they cannot be easily used when smaller than 55 μm pixels are needed due to area limitations. This limitation can only be overcome by moving to smaller CMOS technology nodes or implementing architectures with less hardware. The latter usually implies worse readout efficiency, as was discussed in Chapter 2. Fortunately, the presented solutions can be scaled to fit into smaller pixels using newer CMOS technologies, although at the price of higher non-recurring engineering (NRE) costs. Because presented architectures have been designed using RTL design techniques, they are also more portable to newer technologies than full-custom designs.

The techniques are also not suitable for applications with power requirements of < 100 mW/cm^2 without some modifications. Modifications to the analog front-ends are required, because they contribute almost half of the power consumption in the two presented chips. However, speed (and performance) in digital logic can be traded off for power. Because all power figures presented have been obtained at a frequency of 40 MHz or higher, a reduction of 40 in digital power could be achieved by using a 1 MHz clock or even lower. This would effectively cut down the throughput of the architecture but would also reduce the power consumption below 40 mW/cm^2. Running at lower frequency makes it possible to reduce the power supply from 1.5 to 1.2 V thus yielding an extra reduction in power by 36 %.

7.4 Future outlook

Performance requirements for hybrid pixel chips are constantly increasing. This means more complex and better performing readout chips are required. Even though the architectural optimization offers some possibilities for improving the performance, eventually newer CMOS nodes must be used to meet the performance requirements. Because the complexity of the design increases when moving to newer CMOS technologies, by making a readout chip more programmable, several projects could share a common readout chip architecture and program it to their specific application. In this way, the projects could combine their design resources for shorter design turnaround times.

The usage of programmable processors has increased tremendously in integrated circuits due to CMOS scaling. Domain-specific processors such as DSPs, network processors and baseband processors have also seen increased use. However, pixel readout chips benefit from massive parallelism of pixels, while processors are generally sequential in nature. Thus, a multi-processor solution is called for, if this kind of programmable processor is used. Due to the higher density of integration in processes of 65 nm and beyond, it is feasible to integrate many of these processors even to the periphery of the pixel chip.

Even having many processors to reduce the data rate either by data compression or using trigger does not remove the need to have efficient column and periphery architectures for getting the data into the processors in the first place. Thus, the results of this thesis can be used as a starting point for an investigation into architectures with lower power, higher efficiency and throughput, and lower latency. Because this study has been done using 130 nm CMOS technology, it is expected that possibilities for new solutions emerge when moving to newer technologies with higher transistor densities.

Bibliography

[1] E.R. Fossum. Digital camera system on a chip. *Micro, IEEE*, 18(3):8–15, 1998.

[2] B. Kohn, A.N. Belbachir, and A. Nowakowska. Real-time gesture recognition using bio inspired 3D vision sensor. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 37–42, 2012.

[3] W. S. Wong, G. Anton, R. Ballabriga, G. Blaj, M. Bhnel, M. Campbell, T. Gabor, E. Heijne, X. Llopart, T. Michel, I. Ritter, T. Poikela, P. Sievers, L. Tlustos, and P. Valerio. Electrical measurements of a multi-mode hybrid pixel detector asic for radiation detection. *Journal of Instrumentation*, 7(01):C01056, 2012.

[4] G. Blanchot, M. Chmeissani, A. Daz, F. Daz, J. Fernndez, E. Garca, J. Garca, F. Kainberger, M. Lozano, M. Maiorino, R. Martnez, J.P. Montagne, I. Moreno, G. Pellegrini, C. Puigdengoles, M. Sents, L. Teres, M. Tortajada, and M. Ulln. Dear-mama: A photon counting x-ray imaging project for medical applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 569(1):136 – 139, 2006.

[5] C. Lee and C. Hsieh. A 0.8-V 4096-Pixel CMOS Sense-and-Stimulus Imager for Retinal Prosthesis. *Electron Devices, IEEE Transactions on*, 60(3):1162–1168, March 2013.

[6] The CMS Collaboration. The CMS experiment at the CERN LHC. *Journal of Instrumentation*, 3(08):S08004, 2008.

[7] The ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation*, 3(08):S08003, 2008.

[8] U. Kerzel. The LHCb RICH detectors. *Journal of Physics: Conference Series*, 110(9):092014, 2008.

[9] P. Rodriguez Perez. The LHCb VERTEX LOCATOR performance and VERTEX LOCATOR upgrade. *Journal of Instrumentation*, 7(12):C12008, 2012.

[10] S. Eisenhardt and the LHCb Collaboration. The LHCb Upgrade. *Journal of Physics: Conference Series*, 447(1):012046, 2013.

[11] G. Lutz. *Semiconductor Radiation Detectors.* Springer Link, 2007.

[12] L. Rossi et al. *Pixel detectors - from fundamentals to applications.* Springer, Berlin Heidelberg, 2006.

[13] N. Wermes. Pixel vertex detectors. *arXiv preprint physics/0611075*, 2006.

[14] S. Vähänen, T. Tick, and M. Campbell. Low-cost bump bonding activities at CERN. *Journal of Instrumentation*, 5(11):C11008, 2010.

[15] T. Tick and M. Campbell. TSV processing of Medipix3 wafers by CEA-LETI: a progress report. *Journal of Instrumentation*, 6(11):C11018, 2011.

[16] M. De. Gaspari, J. Alozy, R. Ballabriga, M. Campbell, E. Frojdh, J. Idarraga, S. Kulis, X. Llopart, T. Poikela, P. Valerio, and W. Wong. Design of the analog front-end for the Timepix3 and Smallpix hybrid pixel detectors in 130 nm CMOS technology. *Journal of Instrumentation*, 9(01):C01037, 2014.

[17] H. Spieler. Analog and digital electronics for detectors. *Proceedings of the 2003 ICFA School on Instrumentation*, 2003.

[18] F. Faccio and G. Cervelli. Radiation-induced edge effects in deep submicron CMOS transistors. *Nuclear Science, IEEE Transactions on*, 52(6):2413–2420, 2005.

[19] S. Bonacini, P. Valerio, R. Avramidou, R. Ballabriga, F. Faccio, K. Kloukinas, and A. Marchioro. Characterization of a commercial 65 nm CMOS technology for SLHC applications. *Journal of Instrumentation*, 7(01):P01015, 2012.

[20] M. Garcia-Sciveres, A. Mekkaoui, and D. Ganani. Towards third generation pixel readout chips. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 731(0):83 – 87, 2013. PIXEL 2012.

[21] M. Silvestri, S. Gerardin, F. Faccio, and A. Paccagnella. Single-event gate rupture in 130-nm CMOS transistor arrays subjected to X-ray

irradiation. In *Radiation and Its Effects on Components and Systems (RADECS), 2009 European Conference on*, pages 119–125, Sept 2009.

[22] L. Pierobon, S. Bonacini, F. Faccio, and A. Marchioro. Single-event upset sensitivity of latches in a 90nm dual and triple well CMOS technology. *Journal of Instrumentation*, 6(12):C12011, 2011.

[23] R.E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*, 6(2):200–209, 1962.

[24] V.S. Veeravalli. Fault tolerance for arithmetic and logic unit. In *Southeastcon, 2009. SOUTHEASTCON '09. IEEE*, pages 329–334, 2009.

[25] X. Llopart, R. Ballabriga, M. Campbell, L. Tlustos, and W. Wong. Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 581(1–2):485–494, 2007.

[26] M. Havrnek, T. Hemperek, T. Kishishita, H. Krger, and N. Wermes. Pixel front-end development in 65 nm CMOS technology. *Journal of Instrumentation*, 9(01):C01003, 2014.

[27] A. Hoffman, M. Loose, and V. Suntharalingam. CMOS Detector Technology. *Experimental Astronomy*, 19(1-3):111–134, 2005.

[28] E . W. Bogaart, W. Hoekstra, I. M. Peters, A. Kleimann, and J. T. Bosiers. Very Low Dark Current CCD Image Sensor. *Electron Devices, IEEE Transactions on*, 56(11):2462–2467, 2009.

[29] LHCb Collaboration. LHCb VELO Upgrade Technical Design Report. Technical Report CERN-LHCC-2013-021. LHCB-TDR-013, CERN, Geneva, Nov 2013.

[30] T. Poikela, J. Plosila, T. Westerlund, M. Campbell, M. De Gaspari, X. Llopart, V. Gromov, R. Kluit, M. van Beuzekom, F. Zappon, V. Zivkovic, C. Brezina, K. Desch, Y. Fu, and A. Kruth. Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout. *Journal of Instrumentation*, 9(05):C05013, 2014.

[31] T. Poikela, J. Plosila, T. Westerlund, J. Buytaert, M. Campbell, M. De Gaspari, X. Llopart, K. Wyllie, V. Gromov, R. Kluit, M. van Beuzekom, F. Zappon, V. Zivkovic, C. Brezina, K. Desch, Y. Fu, and A. Kruth. Digital column readout architectures for hybrid pixel detector readout chips. *Journal of Instrumentation*, 9(01):C01007, 2014.

[32] T. Poikela, J. Plosila, T. Westerlund, J. Buytaert, M. Campbell, X. Llopart, R. Plackett, K. Wyllie, M. van Beuzekom, V. Gromov, R. Kluit, F. Zappon, V. Zivkovic, C. Brezina, K. Desch, X. Fang, and A. Kruth. Architectural modeling of pixel readout chips Velopix and Timepix3. *Journal of Instrumentation*, 7(01):C01093, 2012.

[33] M. van Beuzekom, J. Buytaert, M. Campbell, P. Collins, V. Gromov, R. Kluit, X. Llopart, T. Poikela, K. Wyllie, and V. Zivkovic. Velopix ASIC development for LHCb VELO upgrade. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, (0):0, 2013.

[34] V. Gromov, M. van Beuzekom, X. Fang, A. Kruth, R. Kluit, F. Zappon, V. Zivkovic, M. Campbell, T. Poikela, X. Llopart, C. Brezina, and K. Desch. Development and Applications of the Timepix3 Readout Chip. page 046. 20th Anniversary International Workshop on Vertex Detectors, Rust(Austria), June 2011.

[35] P. Valerio, J. Alozy, S. Arfaoui, R. Ballabriga, M. Benoit, S. Bonacini, M. Campbell, D. Dannheim, M. De Gaspari, D. Felici, S. Kulis, X. Llopart, A. Nascetti, T. Poikela, and W. S. Wong. A prototype hybrid pixel detector ASIC for the CLIC experiment. *Journal of Instrumentation*, 9(01):C01012, 2014.

[36] A. Gabrielli, F. Giorgi, and M. Villa. A high efficiency readout architecture for a large matrix of pixels. *Journal of Instrumentation*, 5(07):c07003, 2010.

[37] D. Arutinov, M. Barbero, R. Beccherle, V. Buscher, G. Darbo, R. Ely, D. Fougeron, M. Garcia-Sciveres, D. Gnani, T. Hemperek, M. Karagounis, R. Kluit, V. Kostyukhin, A. Mekkaoui, M. Menouni, J. D. Schipper, and N. Wermes. Digital architecture and interface of the new ATLAS Pixel Front-End IC for upgraded LHC luminosity. In *Nuclear Science Symposium Conference Record, 2008. NSS '08. IEEE*, pages 1923–1928, Oct 2008.

[38] B. Meier. CMS pixel detector with new digital readout architecture. *Journal of Instrumentation*, 6(01):C01011, 2011.

[39] S. Heuvelmans and M. Boerrigter. A pixel read-out architecture implementing a two-stage token ring, zero suppression and compression. *Journal of Instrumentation*, 6(01):C01093, 2011.

[40] E. Conti, J. Christiansen, P. Placidi, and S. Marconi. Pixel chip architecture optimization based on a simplified statistical and analytical model. *Journal of Instrumentation*, 9(03):C03011, 2014.

[41] S. Marconi, E. Conti, P. Placidi, J. Christiansen, and T. Hemperek. The RD53 Collaboration's SystemVerilog-UVM Simulation Framework and its General Applicability to Design of Advanced Pixel Readout Chips. *JINST*, 9(10):P10005, 2014.

[42] R. Ballabriga, J. Alozy, G. Blaj, M. Campbell, M. Fiederle, E. Frojdh, E. H. M. Heijne, X. Llopart, M. Pichotka, S. Procz, L. Tlustos, and W. Wong. The Medipix3RX: a high resolution, zero dead-time pixel detector readout chip allowing spectroscopic imaging. *Journal of Instrumentation*, 8(02):C02016, 2013.

[43] X. LLopart, M. Campbell, R. Dinapoli, D. San Segundo, and E. Pernigotti. Medipix2: a 64-k Pixel Readout Chip With 55-um Square Elements Working in Single Photon Counting Mode . *Transactions on Nuclear Science, IEEE*, 49(5):2279–2283, October 2002.

[44] R. Dinapoli, A. Bergamaschi, S. Cartier, D. Greiffenberg, I. Johnson, J. H. Jungmann, D. Mezza, A. Mozzanica, B. Schmitt, X. Shi, and G. Tinti. MÖNCH, a small pitch, integrating hybrid pixel detector for X-ray applications. *Journal of Instrumentation*, 9(05):C05015, 2014.

[45] M. Noy, G. Aglieri Rinella, A. Cotta Ramusino, M. Fiorini, P. Jarron, J. Kaplon, A. Kluge, E. Martin, M. Morel, L. Perktold, K. Poltorak, and P. Riedler. Characterisation of the NA62 GigaTracker End of Column Demonstrator Hybrid Pixel Detector. *Journal of Instrumentation*, 6(11):C11025, 2011.

[46] G. Mazza, D. Calvo, P. De Remigis, T. Kugathasan, M. Mignone, A. Rivetti, L. Toscano, R. Wheadon, and A. Bonacini. A CMOS 0.13 μm Silicon Pixel Detector Readout ASIC for the PANDA experiment. *Journal of Instrumentation*, 7(02):C02015, 2012.

[47] R. Dinapoli et al. An analog front-end in standard 0.25 μm CMOS for silicon pixel detectors in ALICE and LHCb. *Proceedings of the Sixth Workshop on Electronics for LHC experiments*, 2000.

[48] H. Chr. Kaestli, M. Barbero, W. Erdmann, Ch. Hoermann, R. Horisberger, D. Kotlinski, and B. Meier. Design and Performance of the CMS Pixel Detector Readout Chip. *Nucl. Instrum. Meth.*, pages 188–194, 2006.

[49] I. Peric, L. Blanquart, G. Comes, P. Denes, K. Einsweiler, P. Fischer, E. Mandelli, and G. Meddeler. The FEI3 readout chip for the ATLAS pixel detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 565(1):178 – 187, 2006.

[50] The Open SystemC Initiative. SystemC. SystemC webpage. Accessed on 1 June 2014.

[51] F. Ghenassia et al. *Transaction level modeling with SystemC.* Springer, Dordrecht, 2005.

[52] M. Koyanagi, T. Nakamura, Y. Yamada, H. Kikuchi, T. Fukushima, T. Tanaka, and H. Kurino. Three-dimensional integration technology based on wafer bonding with vertical buried interconnections. *Electron Devices, IEEE Transactions on*, 53(11):2799–2808, 2006.

[53] E. Mandelli, L. Blanquart, P. Denes, K. Einsweiler, R. Marchesini, G. Meddeler, M. Ackers, P. Fischer, G. Comes, and I. Peric. Digital column readout architecture for the ATLAS pixel 0.25 μm front end IC. *Nuclear Science, IEEE Transactions on*, 49(4):1774–1777, Aug 2002.

[54] D. C. Christian, J. A. Appel, G. Cancelo, J. Hoff, S. Kwan, et al. FPIX2 : A Radiation-hard pixel readout chip for BTeV. *Nucl.Instrum.Meth.*, A473:152–156, 2001.

[55] W. J. Dally and R. C. Harting. *Digital Design: A Systems Approach.* Cambridge University Press, 2012.

[56] B.K. Hall, J.A. Appel, G. Cardoso, D. Christian, J. Hoff, S. Kwan, A. Mekkaoui, R. Yarema, and S. Zimmermann. Development of a readout technique for the high data rate BTeV pixel detector at Fermilab. In *Nuclear Science Symposium Conference Record, 2001 IEEE*, volume 1, pages 90–93 vol.1, Nov 2001.

[57] G. Dellacasa, F. Marchetto, G. Mazza, A. Rivetti, S. Martoiu, et al. Pixel read-out architectures for the NA62 gigatracker. pages 85–89, 2008.

[58] W. Buttinger. The ATLAS Level-1 Trigger System. *Journal of Physics: Conference Series*, 396(1):012010, 2012.

[59] L. Pinsky, Son Minh Hoang, J. Idarraga-Munoz, M. Kroupa, N. Stoffle, A. Bahadori, E. Semones, J. Jakubek, Z. Vykydal, D. Turecek, S. Pospisil, H. Kitamura, and S. Kodaira. Summary of the first year of medipix-based space radiation monitors on the ISS. In *Aerospace Conference, 2014 IEEE*, pages 1–8, March 2014.

[60] G. F. Knoll. *Radiation Detection and Measurement.* 4 edition, 2010.

[61] O. Brun and J. Garcia. Analytical Solution of Finite Capacity M/D/1 Queues. *Journal of Applied Probability*, 37(4):pp. 1092–1098, 2000.

[62] Aung Myat Thu Linn, Do Anh Tuan, Chen Shoushun, and Yeo Kiat Seng. Adaptive priority toggle asynchronous tree arbiter for AER-based image sensor. In *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*, pages 66–71, Oct 2011.

[63] G. Aglieri, C. Cavicchioli, P. L. Chalmet, N. Chanlek, A. Collu, P. Giubilato, H. Hillemanns, A. Junique, M. Keil, D. Kim, J. Kim, T. Kugathasan, A Lattuca, M. Mager, C. A. Marin Tobon, D. Marras, P. Martinengo, S. Mattiazzo, G. Mazza, H. Mugnier, L. Musa, D. Pantano, C. Puggioni, J. Rousset, F Reidt, P. Riedler, S. Siddhanta, W. Snoeys, G. Usai, J. W. van Hoorne, P. Yang, and J. Yi. Monolithic active pixel sensor development for the upgrade of the ALICE inner tracking system. *Journal of Instrumentation*, 8(12):C12041, 2013.

[64] R. Chandra and O. Sinnen. Improving application performance with hardware data structures. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–4, April 2010.

[65] E. Caspi. *Design Automation for Streaming Systems.* PhD thesis, 2005.

[66] J. Plosila, P. Liljeberg, and J. Isoaho. Pipelined on-chip bus architecture with distributed self-timed control. In *Signals, Circuits and Systems, 2003. SCS 2003. International Symposium on*, volume 1, pages 257–260 vol.1, 2003.

[67] P. Liljeberg, J. Plosila, and J. Isoaho. Self-timed ring architecture for SOC applications. In *SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip]*, September 2003.

[68] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *Networking, IEEE/ACM Transactions on*, 4(3):375–385, Jun 1996.

[69] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *Communications, IEEE Transactions on*, 47(8):1260–1267, Aug 1999.

[70] J. Rexford, J. Hall, and K.G. Shin. A router architecture for real-time communication in multicomputer networks. *Computers, IEEE Transactions on*, 47(10):1088–1101, Oct 1998.

[71] P. Fischer. Design considerations for pixel readout chips. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators,*

Spectrometers, Detectors and Associated Equipment, 501(1):175 – 182, 2003. Proceedings of the 10th International Workshop on Vertex Detectors.

[72] C. Fallin, X. Yu, G. Nazario, and O. Mutlu. A high-performance hierarchical ring on-chip interconnect with low-cost routers, 2011. SAFARI Technical Report.

[73] V. C. Hamacher and H. Jiang. Hierarchical ring network configuration and performance modeling. *IEEE Trans. Comput.*, 50(1):1–12, January 2001.

[74] L. Blanquart, J. Richardson, K. Einsweiler, P. Fischer, E. Mandelli, G. Meddeler, and I. Peric. FE-I2: a front-end readout chip designed in a commercial 0.25 μm process for the ATLAS pixel detector at LHC. *Nuclear Science, IEEE Transactions on*, 51(4):1358–1364, Aug 2004.

[75] T. Hemperek, D. Arutinov, M. Barbero, R. Beccherle, G. Darbo, S. Dube, D. Elledge, D. Fougeron, M. Garcia-Sciveres, D. Gnani, V. Gromov, M. Karagounis, R. Kluit, A Kruth, A Mekkaoui, M. Menouni, J.D. Schipper, and N. Wermes. Digital architecture of the new ATLAS pixel chip FE-I4. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, pages 791–796, Oct 2009.

[76] R. Dinapoli, A. Bergamaschi, B. Henrich, R. Horisberger, I. Johnson, A. Mozzanica, E. Schmid, B. Schmitt, A. Schreiber, X. Shi, and G. Theidel. Eiger: Next generation single photon counting detector for x-ray applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 650(1):79 – 83, 2011. International Workshop on Semiconductor Pixel Detectors for Particles and Imaging 2010.

[77] W. S. Wong. *A Hybrid Pixel Detector ASIC with Energy Binning for Real-Time, Spectroscopic Dose Measurements.* PhD thesis, Mid Sweden University, Sundsvall, Sweden, 2012.

[78] P. Pangaud, S. Basolo, B. Chantepie, J. C Clemens, P. Delpierre, B. Dinkespiler, M. Menouni, A. Bonissent, F. Debarbieux, and C. Morel. First results of XPAD3, a new photon counting chip for X-ray CT-scanner with energy discrimination. In *Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE*, volume 1, pages 14–18, Oct 2007.

[79] E. Conti, P. Placidi, J. Christiansen, and L. Servoli. *Design of Dedicated Electronic Systems for the Readout of Pixel Radiation Sensors.* PhD thesis, Perugia U., Oct 2014.

[80] L. Cai and D. Gajski. Transaction Level Modeling: An Overview. *First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 1, 2003.

[81] M. E. Conway. Design of a separable transition-diagram compiler. *Commun. ACM*, 6(7):396–408, July 1963.

[82] IEEE 1800-2012: SystemVerilog (SV). Accellera systems initiative, 2012. Accessed on 9 July 2014.

[83] W. Snyder et al. Verilator. Free open-source Verilog simulator. Verilator homepage. Accessed on 8 July 2014.

[84] H. Zhang. An investigation of the relationships between lines of code and defects. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 274–283, Sept 2009.

[85] C. A. Chung. *Simulation Modelling Handbook A Practical Approach.* 2004.

[86] A. V. Gafarian, C. J. Ancker, Jr., and T. Morisaku. The problem of the initial transient in digital computer simulation. In *Proceedings of the 76 Bicentennial Conference on Winter Simulation*, WSC '76, pages 49–51. Winter Simulation Conference, 1976.

[87] R. Ballabriga, M. Campbell, E.H.M. Heijne, X. Llopart, and L. Tlustos. The Medipix3 Prototype, a Pixel Readout Chip Working in Single Photon Counting Mode with Improved Spectrometric Performance. *Nuclear Science Symposium Conference Record, IEEE*, 6:3557–3561, 2006.

[88] W. Wong, R. Ballabriga, M. Campbell, X. Llopart, and L. Tlustos. Counter architectures for a single photon-counting pixel detector such as medipix3. *AIP Conference Proceedings*, 958(1), 2007.

[89] A. Ajane, P.M. Furth, E.E. Johnson, and R.L. Subramanyam. Comparison of binary and LFSR counters and efficient LFSR decoding algorithm. In *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pages 1–4, Aug 2011.

[90] K. Kuusilinna V. Lahtinen, E. Salminen and T. D. Hämäläinen. Bus structures in Network-on-Chips. In J. Isoaho J. Nurmi, H. Tenhunen and A. Jantch, editors, *Interconnect-Centric Design for Advanced SOC and NOC*, pages 207–230. Kluwer Academic Publishers, 2004.

[91] T. Sakurai. Approximation of wiring delay in MOSFET LSI. *Solid-State Circuits, IEEE Journal of*, 18(4):418–426, Aug 1983.

[92] R. Singhal, Gwan Choi, and R.N. Mahapatra. Data handling limits of on-chip interconnects. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(6):707–713, June 2008.

[93] Q.K. Zhu and A. Wu. Design of bi-directional bus nets. In *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, volume 2, pages 495–498 Vol.2, Dec 2003.

[94] O. Torheim. *Design and implementation of fast and sparsified readout for Monolithic Active Pixel Sensors.* PhD thesis, The University of Bergen, Bergen, Norway, 2010.

[95] H. Kaeslin. *Digital Integrated Circuit Design - From VLSI Architectures to CMOS Fabrication.* Cambridge University Press, 2008.

[96] *FIFO architecture, Functions and Applications.* Texas Instruments, November 1999.

[97] A. Roy J. Xu and M. H. Chowdhury. Analysis of Power Consumption and BER of Flip-flop Based Interconnect Pipelining. *Proceedings of Design, Automation and Test in Europe (DATE)*, pages 1–6, 2007.

[98] X. Llopart. *Design and characterization of 64K Pixels Chips Working in Single Photon Processing Mode.* PhD thesis, Mid Sweden University, Sundsvall, Sweden, 2007.

[99] A.S. Tremsin, J.V. Vallerga, J.B. McPhate, O.H.W. Siegmund, and R. Raffanti. High Resolution Photon Counting With MCP-Timepix Quad Parallel Readout Operating at ¿ 1 kHz Frame Rates. *Nuclear Science, IEEE Transactions on*, 60(2):578–585, April 2013.

[100] A. Kiss, J. H. Jungmann, D. F. Smith, and R. M. A. Heeren. Microscope mode secondary ion mass spectrometry imaging with a Timepix detector. *Review of Scientific Instruments*, 84(1):013704–013704–7, Jan 2013.

[101] C. Brezina, Y. Fu, F. Zappon, M. van Beuzekom, M. Campbell, K. Desch, H. van der Graaf, V. Gromov, R. Kluit, X. Llopart, T. Poikela, and V. Zivkovic. GOSSIPO-4: Evaluation of a Novel PLL-Based TDC-Technique for the Readout of GridPix-Detectors. *Nuclear Science, IEEE Transactions on*, 61(2):1007 – 1014, February 2014.

[102] J. Sparsø and S. Furber. *Principles of asynchronous circuit design - A systems perspective.* Kluwer Academic Publishers, Boston/Dordrecth/London, 2002.

[103] D. M. Chapiro. *Globally-asynchronous locally-synchronous systems.* PhD thesis, Stanford Univ., CA., 1984.

[104] Y. Fu, C. Brezina, K. Desch, T. Poikela, X. Llopart, M. Campbell, D. Massimiliano, V. Gromov, R. Kluit, M. van Beauzekom, F. Zappon, and V. Zivkovic. The charge pump PLL clock generator designed for the 1.56 ns bin size time-to-digital converter pixel array of the Timepix3 readout ASIC. *Journal of Instrumentation*, 9(01):C01052, 2014.

[105] CERN and Nikhef. Timepix3 readout (SPDR). `https://lbtwiki.cern.ch/bin/view/VELO/Timepix3`. Accessed on 7.1.2015.

[106] Youngkyun Jeong, Yoon-Chul Choi, Eun-Ji Choi, Seogheon Ham, Kee-Won Kwon, Young-Hyun Jun, and Jung-Hoon Chun. 0.37mW/Gb/s low power SLVS transmitter for battery powered applications. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 1955–1958, May 2012.

[107] A. Nomerotski, J. Buytart, P. Collins, R. Dumps, E. Greening, M. John, A. Mapelli, A. Leflat, Y. Li, G. Romagnoli, and B. Verlaat. Evaporative CO_2 cooling using microchannels etched in silicon for the future LHCb vertex detector. *Journal of Instrumentation*, 8(04):P04004, 2013.

[108] M. Nikolic and M. Katona. Improve the automatic clock gating insertion in asic synthesis process using optimal enable function selection. In *Circuits and Systems for Communications (ECCSC), 2010 5th European Conference on*, pages 131–134, Nov 2010.

[109] S. Wimer and I. Koren. Design flow for flip-flop grouping in data-driven clock gating. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(4):771–778, April 2014.

[110] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sept 1952.

[111] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[112] T. Bird et al. VP Simulation and Track Reconstruction. Technical Report LHCb-PUB-2013-018. CERN-LHCb-PUB-2013-018, CERN, Geneva, Oct 2013.

[113] J. Lechner and M. Lampacher. Protecting pipelined asynchronous communication channels against single-event upsets. *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, pages 480–481, 2012.

[114] V. Gromov et al. Development of a lower power 5.12 Gbps Data Serializer and Wireline Transmitter circuit for VeloPix chip. *JINST*, 2015. Unpublished.

[115] M. Clemencic, G. Corti, S. Easo, C. R. Jones, S. Miglioranzi, M. Pappagallo, P. Robbe, and the LHCb Collaboration. The LHCb Simulation Application, Gauss: Design, Evolution and Experience. *Journal of Physics: Conference Series*, 331(3):032023, 2011.

[116] S. Mitra and E.J. McCluskey. Word-voter: a new voter design for triple modular redundant systems. In *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*, pages 465–470, 2000.

[117] B. Baykant Alagoz. Hierarchical triple-modular redundancy (H-TMR) network for digital systems. *CoRR*, abs/0902.0241, 2009.

[118] T. Calin, M. Nicolaidis, and R. Velazco. Upset hardened memory design for submicron CMOS technology. *Nuclear Science, IEEE Transactions on*, 43(6):2874–2878, Dec 1996.

ISBN 978-952-12-3235-0
ISSN 1239-1883

Tuomas Poikela

Readout Architectures for Hybrid Detector Readout Chips

Tuomas Poikela

Readout Architectures for Hybrid Pixel Detector Readout Chips